NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

APL TUTOR:

AN ON-LINE INSTRUCTIONAL FACILITY

by

Katherine S. Lanes

December 1983

Thesis Advisor: R. R. Read

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) APL TUTOR: An On-Line Instructional Facility		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis December 1983 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Katherine A. Lanes		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE December 1983 13. NUMBER OF PAGES 200
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

APL, programming language, tutorial, CAI, CAL

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis describes a set of APL programs which enable a student to learn A Programming Language (APL) by using it. The student needs to know only how to log on to the computer and enter a few simple commands to begin the course.

The basic unit of the TUTOR workspace is the text variable which describes the use of one of seventy-five built-in APL functions. This description is accessible by a HELP function

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20.

without going through an entire lesson. Other functions conduct interactive question-and-answer drill. A MENU function lists the units for student selection of a lesson or information on a symbol. The student can also go through a sequence of lessons in a computer-driven course.

These programs were written on and for the IBM 3033 installation at Naval Postgraduate School, using APL version 4.0 for VM/CMS with IBM 3278 terminals.

Approved for public release; distribution unlimited.

APL TUTOR:
An On-Line Instructional Facility

by

Katherine S. Lanes
Lieutenant, United States Navy
B.A., New Mexico State University, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
December 1983

ABSTRACT

This thesis describes a set of APL programs which enable a student to learn A Programming Language (APL) by using it. The student needs to know only how to log on to the computer and enter a few simple commands to begin the course.

The basic unit of the TUTOR workspace is the text variable which describes the use of one of seventy-five built-in APL functions. This description is accessible by a HELP function without going through an entire lesson. Other functions conduct interactive question-and-answer drill. A MENU function lists the units for student selection of a lesson or information on a symbol. The student can also go through a sequence of lessons in a computer-driven course.

These programs were written on and for the IBM 3033 installation at Naval Postgraduate School, using APL version 4.0 for VM/CMS with IBM 3278 terminals.

TABLE OF CONTENTS

LIST OF FIGURES

I. BACKGROUND

A Programming Language (APL) is an interactive, interpretive computer language designed by Kenneth E. Iverson at Harvard in 1962. Its advantages include the ease of manipulation of multi-dimensional arrays of numbers, the lack of rigorous input/output formatting, and the use of unique symbols to represent a wide variety of built-in functions. Unlike other high-level languages, there is no requirement to write and compile a strictly formatted program before execution. APL may be used in "desk calculator" mode, where a single line of input is accepted and evaluated, returning an immediate response. While it may take quite some time for the user to master APL fully, the beginner can quickly jump in and start experimenting.

The APL TUTOR was designed to enable the beginning APL student to gain some familiarity with the basic concepts of the language without going through a formal course. It consists of a set of APL programs which enable the student to enter simple commands and receive information, questions, and drill about the symbols and functions used in APL. Since at present no information on writing functions is included, it may be used as a supplement to the normal mode of teaching at NPS.

Currently the APL student's resources at NPS consist of the instructor, other students, and texts such as those listed in the bibliography. There also exists a copyrighted IBM drill function, TEACH, found in the APLCOURS workspace in the public library of most APL installations. (At NPS, it can be accessed by entering)LOAD 1 APLCOURS while in APL mode.) This function, and a sub-function, EASYDRILL, provide simple drill questions for calculator functions

selected by the student. This workspace does not, however, provide any information concerning the correct use or syntax of any functions.

The specific audience targeted is the first to third quarter NPS students in the Operations Analysis curriculum, who are currently taught APL in one-hour weekly labs as part of three probability and statistics courses. Many, if not most, of these students arrive at NPS from six weeks to six months before their first quarter. It would be to their advantage to get a 'head start' on learning APL, but there is no separate course in this language as there is in FORTRAN or PASCAL, for example.

A basic requirement for the TUTOR, then, is that it be comprehensible to someone who has never seen APL before. In fact, the "desk calculator" functions should be understandable to someone who has never even used a computer before. This means that some of the beginning lessons may seem too simplistic to the student who has some programming experience. However, the more advanced student can easily skim through this primary material and progress to the more sophisticated concepts. Intermediate mathematics including concepts of linear algebra is the only background information that is assumed.

II. USING APL TUTOR

The APL TUTOR workspace should be available in one of the public libraries at NPS. Instructions for loading and copying it are on the first page of Appendix A. If it is not available in the public libraries, it can be obtained from Professor R. R. Read upon request.

The APL TUTOR workspace contains sixteen functions which administer the course. Their interrelationship is shown in Figure B.1. In this diagram, the ovals indicate functions which are called by the student, while rectangles indicate functions which are called internally. The procedure of each function is outlined in Figures B.4 through B.21. Further information can be found in the comment lines of each program in Appendix B, part B.

The actual text of the course is contained in APL variables which are displayed by the functions SHOW and RUN. The text of each variable can be found in Appendix B, part C.2. Also in Appendix B (part C.1) are several global variables which are used in various programs. Most important of these is MAT, which provides information on each text variable, such as the symbol it pertains to, the function type (Monadic/Dyadic/Neither), the type of arguments it takes (Numeric, character(K), Either, Boolean), the rank of the arguments permitted (Scalar, Vector, matrix(X), Any), and a sequential lesson number. The other major global variable is CUES, which consists of a number of questions that can be called upon by the ASK function.

Before commencing the APL TUTOR course, the student needs to have, as a minimum, the instructions given on the first page of Appendix A. The other information the student will need is displayed by the TUTOR, and is also included in

Appendix A, which should be given to all students who will be using TUTOR. The variable HOW is displayed each and every time the workspace TUTOR is loaded by the student, and can also be called up at any time by entering HOW. The other variables in Appendix A are displayed by the START function. Most may be called up by the student at any time, as noted.

Figure B.1 shows the variety of ways the student can access and use the information in TUTOR. The beginner enters START to get the basic background information needed to complete the course. START instructs the student to enter LESSON 100 to begin the course. After completing this lesson, the student merely has to enter LESSON NEXT to receive the instructional units in the order indicated in the variable MAT.

The student who is enrolled in a course using APL may also follow the instructor's guidance concerning lessons to take. The student enters LESSON NNN, where NNN is the lesson number assigned. A more advanced student can explore or review earlier lessons by entering TEACH and then the symbol that he/she is interested in.

The more advanced student may also be interested in the HELP function which displays information without asking questions or providing a drill. The student can enter HELP, followed by the appropriate symbol, or INFO NNN, where NNN is the same as the pertinent lesson number. These lesson numbers can be found by entering MENU and browsing the list displayed. MENU also repeats the instructions for using LESSON and INFO.

Effort has been made to ensure that the programs will not terminate abnormally in response to a student input error. Every input is checked for validity before it is processed. For example, see the function DRILL, page 59, at lines 8 to 17. However, because it is not possible to check

for every possible input or combination of inputs, there may still be some way to cause an error. Instructions to the student in case of an error are included in the BASICS displayed by the START function. The student should have a hard copy of these instructions, as well.

Any input by the student at a time when no input is called for will cause an abnormal termination. This is inherent in the APL interpreter and cannot be avoided by the programmer. However, all the student needs to do in this case is restart by entering one of the elementary commands (HELP, TEACH, LESSON, etc.)

A sample run is included at Appendix C.

III. EXTENDING APL TUTOR

There are many ways in which the APL TUTOR can be expanded to improve its usefulness. Lessons could be included on the many system commands and variables which are used in APL. Auxiliary lessons on groups of functions, e.g. logical functions, and their basic concepts could provide useful additional information. Lessons could be added to describe programming techniques, and amplifying lessons on applications could be provided. Probably any of the lessons could benefit from additional questions to be included in the ASK function. Also, the ASK function itself could be modified to permit more complex questions and answers.

Another type of improvement from the pedagogical point of view would be the inclusion of facilities for measuring and recording students' progress. The student could be tested after every lesson or group of lessons and the scores recorded for the instructors review. Or, the course could be designed to automatically review the student's weak areas and retest before going on. In any case, it might be beneficial to include lessons which review concepts covered by the course so far.

A test and evaluation of the TUTOR would be extremely beneficial in comparing it to the current teaching methods. Students could be tested for their knowledge of APL and those who had used the TUTOR compared to those who had not. Furthermore, different versions of the TUTOR could be tested against each other. Several different texts could be used, or versions with and without the drill functions.

This could lead to a still more sophisticated modification of the course, in combination with programmed student testing. After an initial unit, the testing could be

designed to reveal what type of instruction, e.g. drill or no drill, works best for that individual student. Or, the student could simply be asked his or her preferences in terms of teaching techniques. Then the course would automatically tailor later lessons to the individual.

All of these changes require someone skilled in APL to modify the current programs. These programs have been heavily documented by flow charts and comment lines (Appendix B) in order to make the task of modification easier. Also, specific instructions for certain types of modifications are included in the variable HOWMODS, Appendix B, part D. Any advanced student can personalize a copy of TUTOR with the help of these tips. For more substantial modifications such as those suggested above, the point of contact for the public version of TUTOR at NPS is Professor R. R. Read.

APPENDIX A

USER'S GUIDE TO APL TUTOR

IF YOU HAVE NEVER EVER USED THE COMPUTER BEFORE, START HERE:

1. Go to the Registration and Accounting Office inside Ingersoll rm. 141. Ask for a user number. You will need to tell them the password you want to use.
2. While you are there, pick up a copy of NPS Technical Note VM-01, "User's Guide to VM/CMS at NPS."
3. Follow the instructions in VM-01 for logging on and formatting your disk. If your keyboard has little red symbols on it, you may continue with step 6 below. Otherwise, log off (see VM-01) and continue with step 4 when you are ready.

IF YOU HAVE FORMATTED YOUR DISK, START HERE:

4. Find a free terminal (in Ingersoll room 141 or room 369) that has red symbols on the keyboard.
5. Log on and enter your password.
6. When you see a line beginning CMS..., press the ENTER key.
7. When you see a line beginning R: ..., enter the letters APL.
8. You should see several lines appear ending with the line CLEAR WS. If you don't, stop here and get help!
9. Now look at the bottom center of your screen, below the line. If you see the letters APL, go on to the next step. If you don't, hold down the ALT key (next to the ENTER key) and press the large key at the top right of the main keyboard which has APL ON-OFF in red letters on the front. The letters APL should now appear at the bottom of your screen. This tells the terminal to use the red symbols when you upshift. To use the red symbols on the front of the keys, you must hold down the ALT key instead of the SHIFT key.
10. IF THIS IS THE VERY FIRST TIME YOU ARE USING APL TUTOR, ENTER)LOAD 5 TUTOR. (Remember to use the red parenthesis, third row, far right, not the black parenthesis in the top row.) OR FOLLOW THE DIRECTIONS GIVEN BY YOUR INSTRUCTOR. If you have used TUTOR before, and you followed the directions in steps 12 and 13 below, just enter)LOAD TUTOR.
11. Now just follow the directions which appear on the screen. If no directions appear, and all you see is a line which says SAVED... and possibly a line which says WSSIZE..., enter HOW to see the directions. To be sure that the directions appear every time, enter START, and follow steps 12 and 13 when you finish your session.
12. If this is the first time you have used TUTOR, enter)WSID TUTOR.
13. When you are ready to quit for the day, enter)SAVE.
14. When you see the time, date, and TUTOR, enter)OFF. This will log you off the computer completely. (For more advanced students:)OFF HOLD will return you to CMS.)

```
      HOW

YOU MAY USE THE APL TUTOR IN THREE WAYS;
(1)  ENTER;   HELP
     TO SELECT THE SYMBOLS THAT YOU WANT INFORMATION ABOUT,
(2)  ENTER;   TEACH
     TO SELECT THE SYMBOLS THAT YOU WANT INFORMATION AND DRILL ON,
(3)  ENTER;   MENU
     TO SEE A LIST OF SYMBOLS AND TOPICS,

IF YOU HAVE NEVER USED THE APL TUTOR BEFORE, ENTER;   START

TO SEE THESE INSTRUCTIONS AGAIN AT ANY TIME, ENTER;   HOW
      .
      .
```

```
      INTRO

WELCOME TO THE APL TUTOR.

THE PURPOSE OF THIS WORKSPACE IS TO INTRODUCE YOU TO
   'A PROGRAMMING LANGUAGE' BY DESCRIBING THE FUNCTIONS OF THE
   MANY SPECIAL APL SYMBOLS, AND BY OUTLINING THE PROCEDURES FOR
   DESIGNING YOUR OWN FUNCTIONS.

      .


      BACKGROUND

THE APL TUTOR ASSUMES YOU HAVE HAD LINEAR ALGEBRA AND TRIGINOMETRY.
   CALCULUS IS NOT NECESSARY.
IF YOU HAVE NEVER HAD ANY COMPUTER PROGRAMMING BEFORE, DON'T WORRY.
   YOU CAN START USING APL RIGHT AWAY, AS A SUPER-SOPHISTICATED
   CALCULATOR WITH MANY BUILT-IN FUNCTIONS.
IF YOU HAVE STUDIED OTHER COMPUTER PROGRAMMING LANGUAGES, RELAX.
   APL IS NOT LIKE ANY OF THE OTHER MAJOR HIGH-LEVEL LANGUAGES.
   YOU CAN FORGET ABOUT DATA TYPES, INPUT/OUTPUT FORMATTING, AND
   MANY OF THE OTHER TEDIOUS DETAILS OF FORTRAN, PASCAL, ETC.
   AFTER YOU ARE SUFFICIENTLY FAMILIAR WITH THE CALCULATOR MODE
   OF APL, YOU CAN LEARN TO DEFINE YOUR OWN FUNCTIONS WITH EASE.

      .
```

BASICS

HERE IS SOME BASIC INFORMATION YOU WILL NEED TO KNOW IN ORDER TO UNDERSTAND THE APL TUTOR LESSONS.

MONADIC AND DYADIC FUNCTIONS

YOU ARE ALREADY FAMILIAR WITH SYMBOLS WHICH REPRESENT ARITHMETIC FUNCTIONS, SUCH AS + OR ÷. APL USES THESE SYMBOLS AND MANY OTHERS TO REPRESENT A VARIETY OF FUNCTIONS.

MANY FUNCTIONS, SUCH AS +, REQUIRE TWO ARGUMENTS, THAT IS, TWO INPUT NUMBERS. IN APL, THESE ARE CALLED DYADIC FUNCTIONS, AND THE SYMBOL IS PLACED BETWEEN THE ARGUMENTS; FOR EXAMPLE, 3+4.

OTHER FUNCTIONS, SUCH AS LN (NATURAL LOG), REQUIRE ONLY ONE ARGUMENT. THESE MONADIC FUNCTION SYMBOLS ARE PLACED TO THE LEFT OF THE DATA WHICH THEY ARE TO OPERATE ON. FOR EXAMPLE, LN 3 IN APL IS ⍟3.

DATA TYPES

APL DISTINGUISHES ONLY TWO TYPES OF DATA; NUMERIC AND CHARACTER.

VERY SIMPLY, CHARACTER DATA ARE ENCLOSED IN QUOTES (') WHEN ENTERED. '2' IS CHARACTER DATA; 2 IS NUMERIC DATA. SOME FUNCTIONS WILL OPERATE ON BOTH TYPES OF DATA, SOME ONLY ON NUMERIC.

ARRAYS

THE GREATEST STRENGTH OF APL LIES IN ITS ABILITY TO TAKE AN ENTIRE ARRAY OF NUMBERS AS A SINGLE ARGUMENT. THUS TWO MATRIXES CAN BE ADDED BY ENTERING SIMPLY A+B, WITH NO SUBSCRIPTS, LOOPS, ETC.

THIS MAKES IT IMPORTANT TO KNOW THE RANK (THE NUMBER OF DIMENSIONS) OF DATA IN USE. A SINGLE NUMBER IS NORMALLY A SCALAR (RANK 0). A SERIES OF NUMBERS IS A VECTOR, A ONE-DIMENSIONAL ARRAY (RANK 1).

NUMBERS CAN ALSO BE ARRANGED IN ROWS AND COLUMNS, TO MAKE A MATRIX (RANK 2). MATRIXES CAN BE 'STACKED' TO MAKE UP THE PAGES OF A THREE-DIMENSIONAL ARRAY, OFTEN CALLED A BOOK (RANK 3). IN FACT, THERE IS NO LIMIT TO THE NUMBER OF DIMENSIONS IN AN APL ARRAY.

THE LENGTH OF THE DIMENSIONS IS ALSO EFFECTIVELY UNLIMITED. LENGTH

REFERS TO THE NUMBER OF ELEMENTS IN A DIMENSION, FOR EXAMPLE, THE NUMBER OF ROWS. LENGTH MAY EVEN BE 0.

IN THIS TUTORIAL, A SCALAR MAY BE REFERRED TO AS S, L, OR R, A VECTOR MAY BE CALLED V, V1, OR V2, WHILE A MATRIX IS M, M1, OR M2. AN ARRAY, WHICH MAY BE A VECTOR, A MATRIX, OR AN ARRAY OF ANY HIGHER DIMENSION, WILL BE LABELLED A, A1, OR A2.

EXAMPLES

WHEN EXAMPLES ARE GIVEN IN THE TEXT OF A FUNCTION DESCRIPTION, THE SYMBOL => IS USED BETWEEN THE EXAMPLE INPUT AND THE EXAMPLE OUTPUT.

FOR EXAMPLE: 2+5 => 7

THIS SYMBOL CAN BE READ AS 'PRODUCES' OR 'RETURNS'.

ERRORS

YOU SHOULD NOT RECEIVE ANY ERROR MESSAGES WHILE USING APL TUTOR. HOWEVER, YOU MAY SEE A PHRASE LIKE ''VALUE ERROR' (WITHOUT QUOTE MARKS) WHEN YOU ARE EXPECTING A NORMAL RESPONSE.

OR IF YOU ACCIDENTALLY HIT THE ENTER KEY WHEN THE COMPUTER IS NOT EXPECTING AN INPUT, THE FUNCTION MAY END ABRUPTLY, LIKE THIS:

```
MENU [6]
```

IF EITHER OF THESE THINGS HAPPEN, BE SURE TO ENTER: → (UPSHIFT ÷) THEN RESTART WITH ANY COMMAND.

HALTING A LESSON

YOU MAY STOP ANY LESSON AT ANY POINT WHERE A QUESTION IS ASKED BY ENTERING: STOP

TO SEE THIS INFORMATION AGAIN AT ANY TIME, ENTER: BASICS

```
HOWTEACH

YOU MAY START A LESSON IN THREE WAYS;

(1)  ENTER;   LESSON NEXTLESSON
     TO START THE LESSON FOLLWING THE LAST ONE YOU COMPLETED.

(2)  ENTER;   LESSON NNN      (WHERE NNN IS A 3-DIGIT NUMBER)
     TO START LESSON NUMBER NNN.
     TO SEE A LIST OF LESSON NUMBERS, ENTER;   MENU

(3)  ENTER;   TEACH
     TO SELECT THE SYMBOL THAT YOU WANT A LESSON ON.

TO SEE THIS INFORMATION AGAIN AT ANY TIME, ENTER;  HOWTEACH
```

APPENDIX B
PROGRAMMER'S GUIDE

A. FLOW CHARTS

Figure B.1 shows the relationship of the various functions within the TUTOR workspace. The first function called by the new student should be START. After that, the student will usually commence a session by calling MENU, TEACH, HELP, LESSON, or INFO. These functions in turn call the other functions of TUTOR as indicated by the arrows. The procedure of these functions is outlined in figures B.2 through B.21.

In APL, there is no difference in the programming of main routines, subroutines, and functions; all are referred to as functions. However, when one function calls another function, the calling function is suspended until the called function is completed. Then the calling function resumes operation, possibly using the result of the called function. Therefore, "return to the calling function, if any" is implied at the end of every program but is not explicit. END, implying return, is shown in the diagrams only in those functions which are normally called by another function.

It should be noted that "room for expansion" has been provided in the SHOW, RUNDRILL, DRILL, and TEST functions. RUNDRILL and DRILL have the capability to display drill questions with matrix arguments. This capacity is not currently being used because the answer must be input in vector form. It was felt that this might confuse the student. However, further development and experimentation might find uses for this type of question, so the matrix form has been left available. TEST and SHOW have calling

lines for the matrix form commented out. There are also comment lines in SHOW to indicate where matrix and higher-level arrays questions would go if they are added in the future. See the section on making modifications for more information.

The four functions at the bottom of figure B.1 are not part of the TUTOR function hierarchy. NEXTLESSON is used to provide a lesson number when the student calls for LESSON NEXTLESSON. ORDERMAT is provided for the benefit of the programmer to rearrange the sequence of lessons in the course. TEST is also provided for the programmer (or student) who wishes to run only the drill section of a lesson. SQUEEZE is a utility called by several functions to delete blanks in a string of characters.

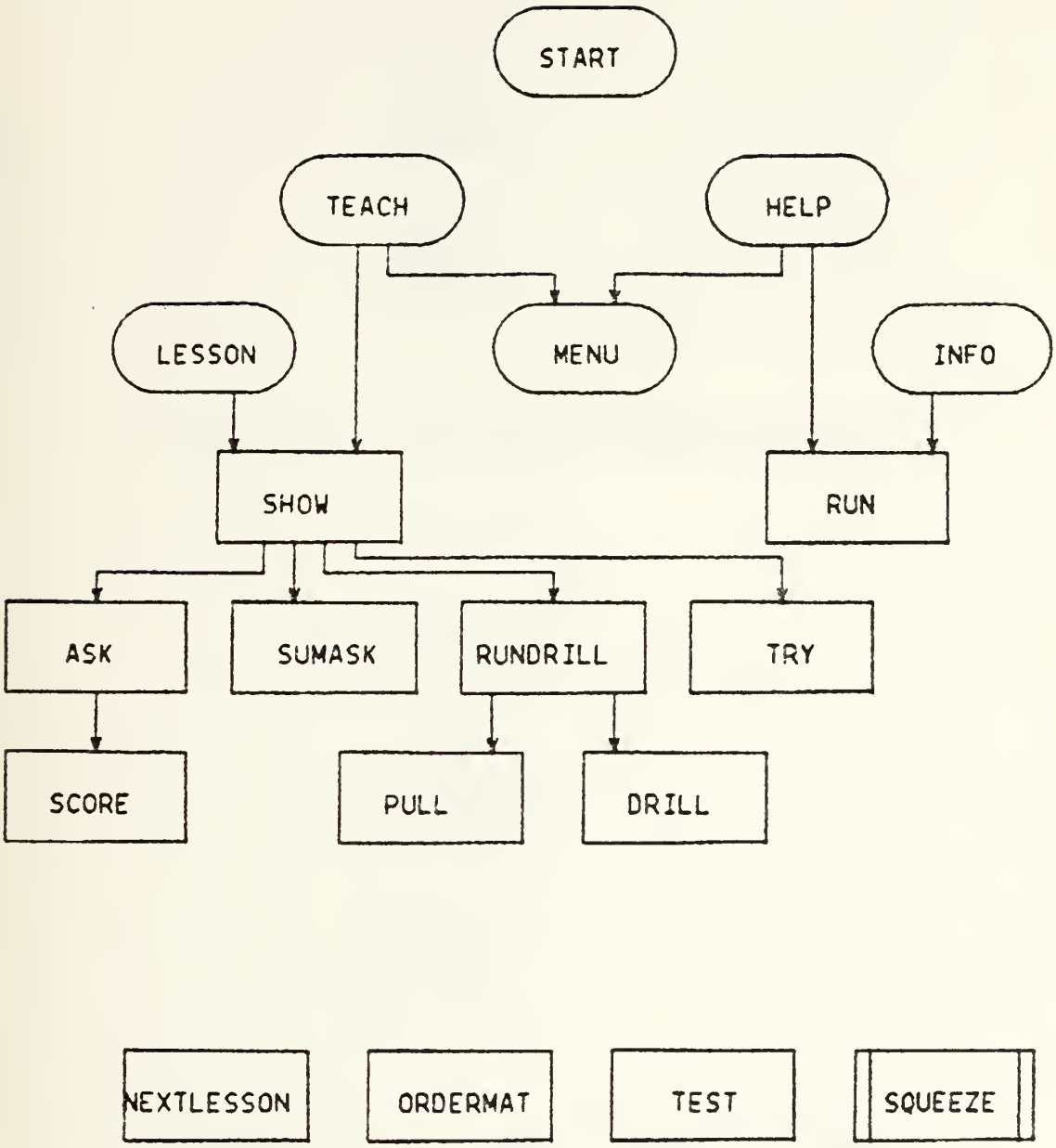

Figure B.1 Interrelationship of TUTOR Functions.

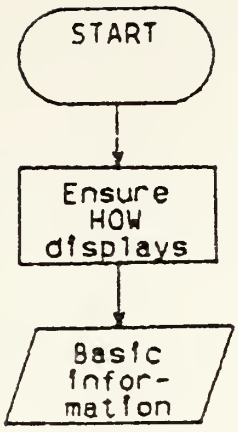

Figure B.2 Procedure of START Function.

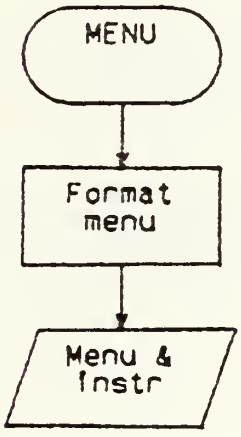

Figure B.3 Procedure of MENU Function.

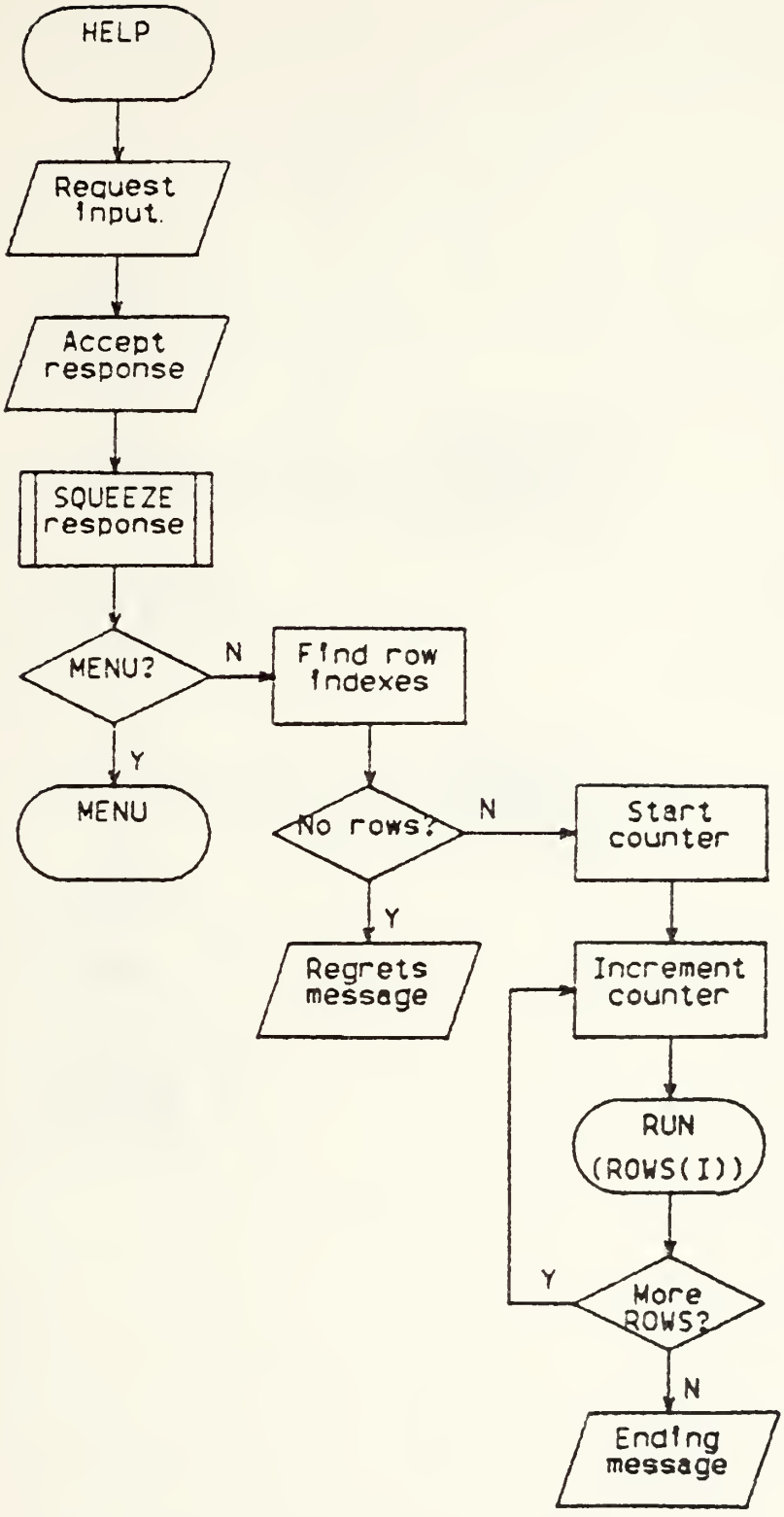

Figure B.4 Procedure of HELP Function.

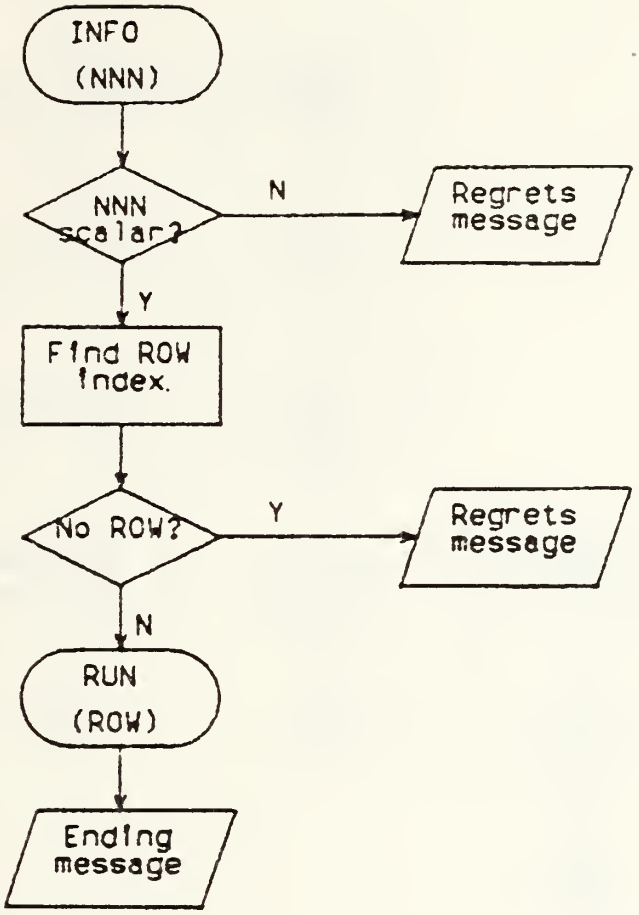

Figure B.5 Procedure of INFO Function.

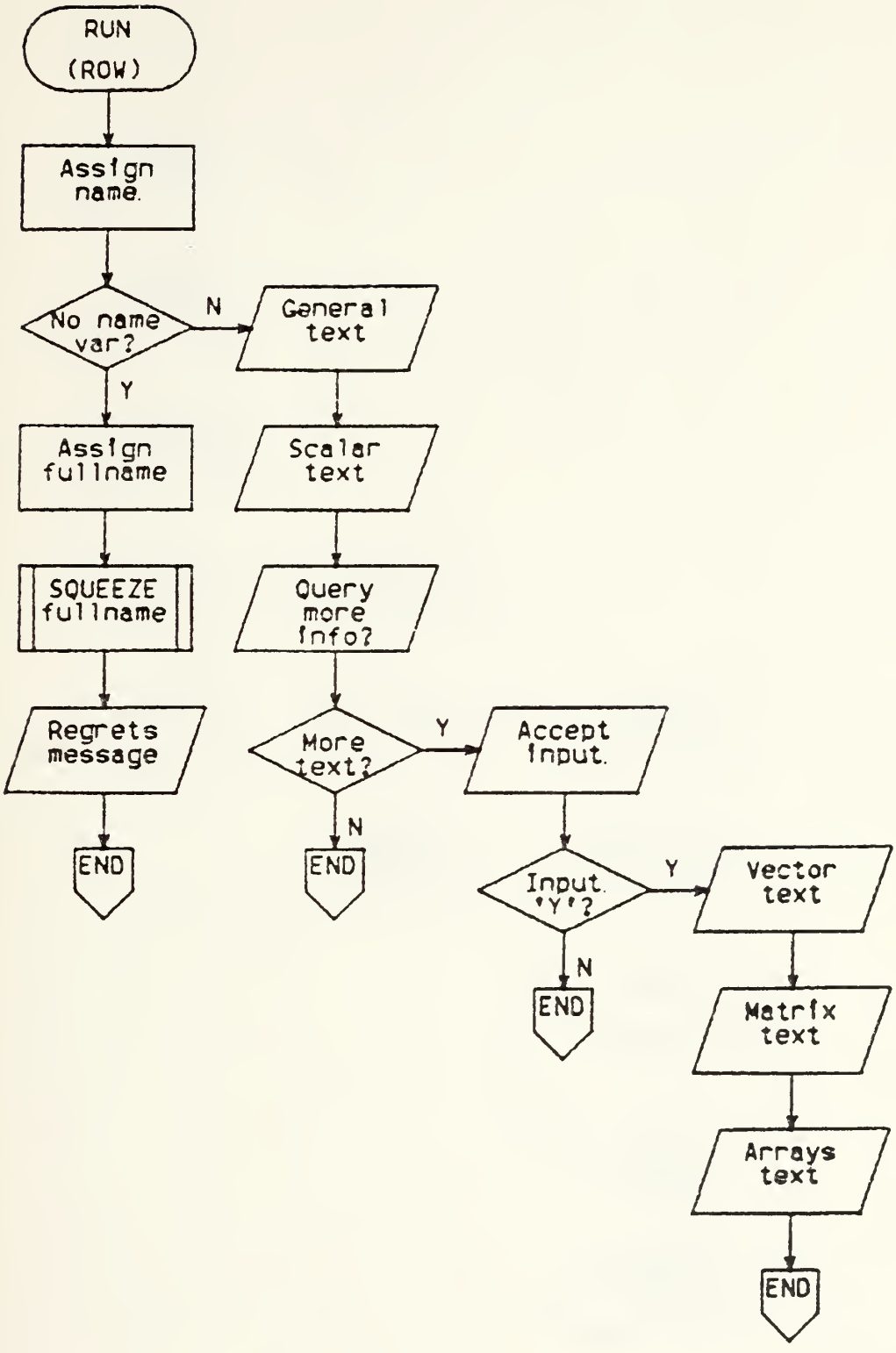

Figure B.6 Procedure of RUN Function.

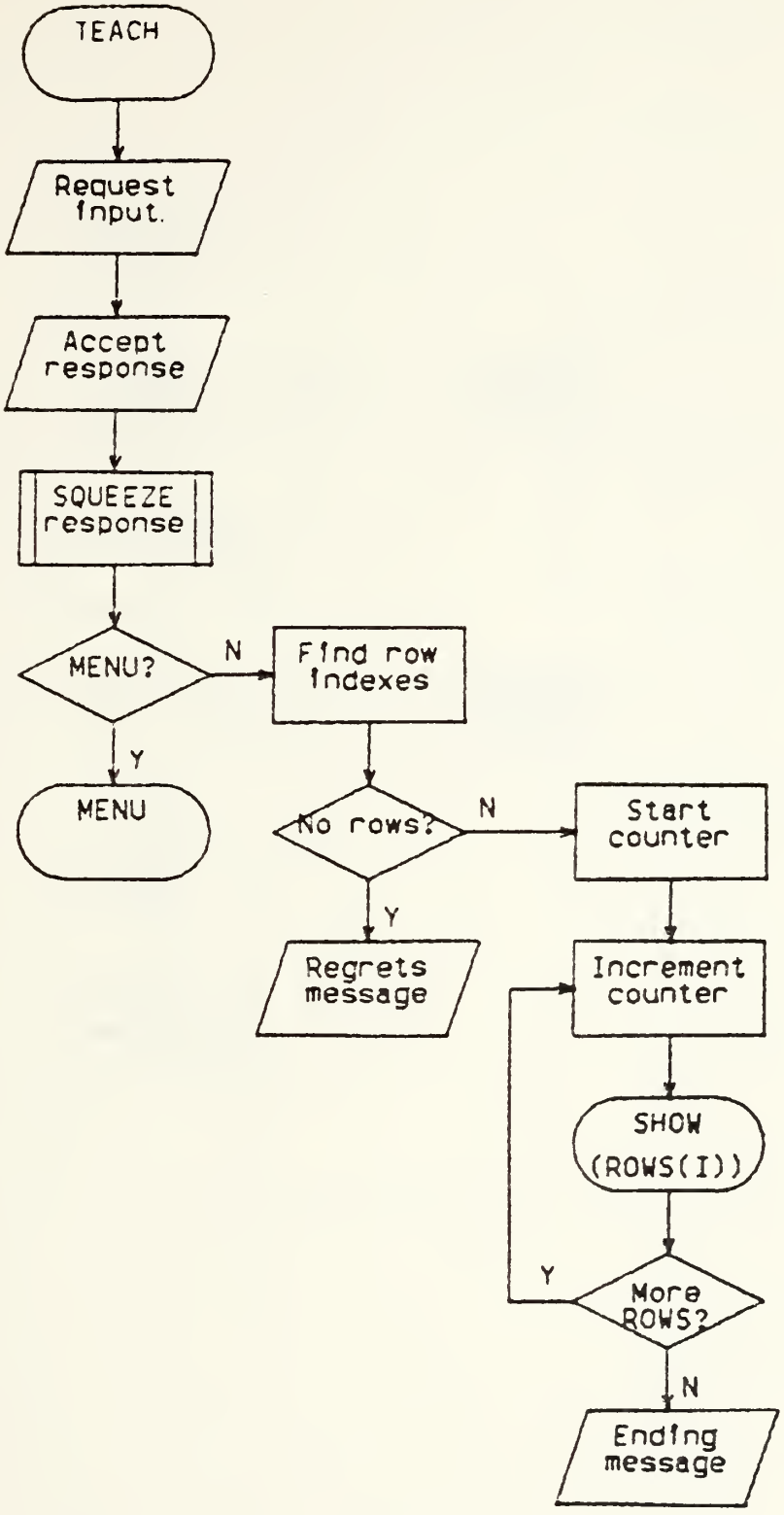

Figure B.7 Procedure of TEACH Function.

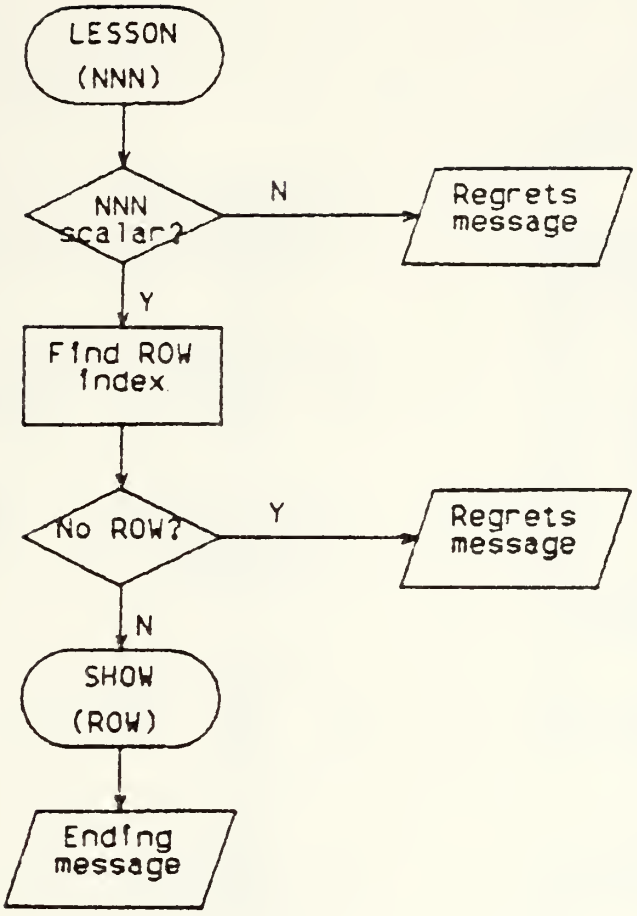

Figure B.8 Procedure of LESSON Function.

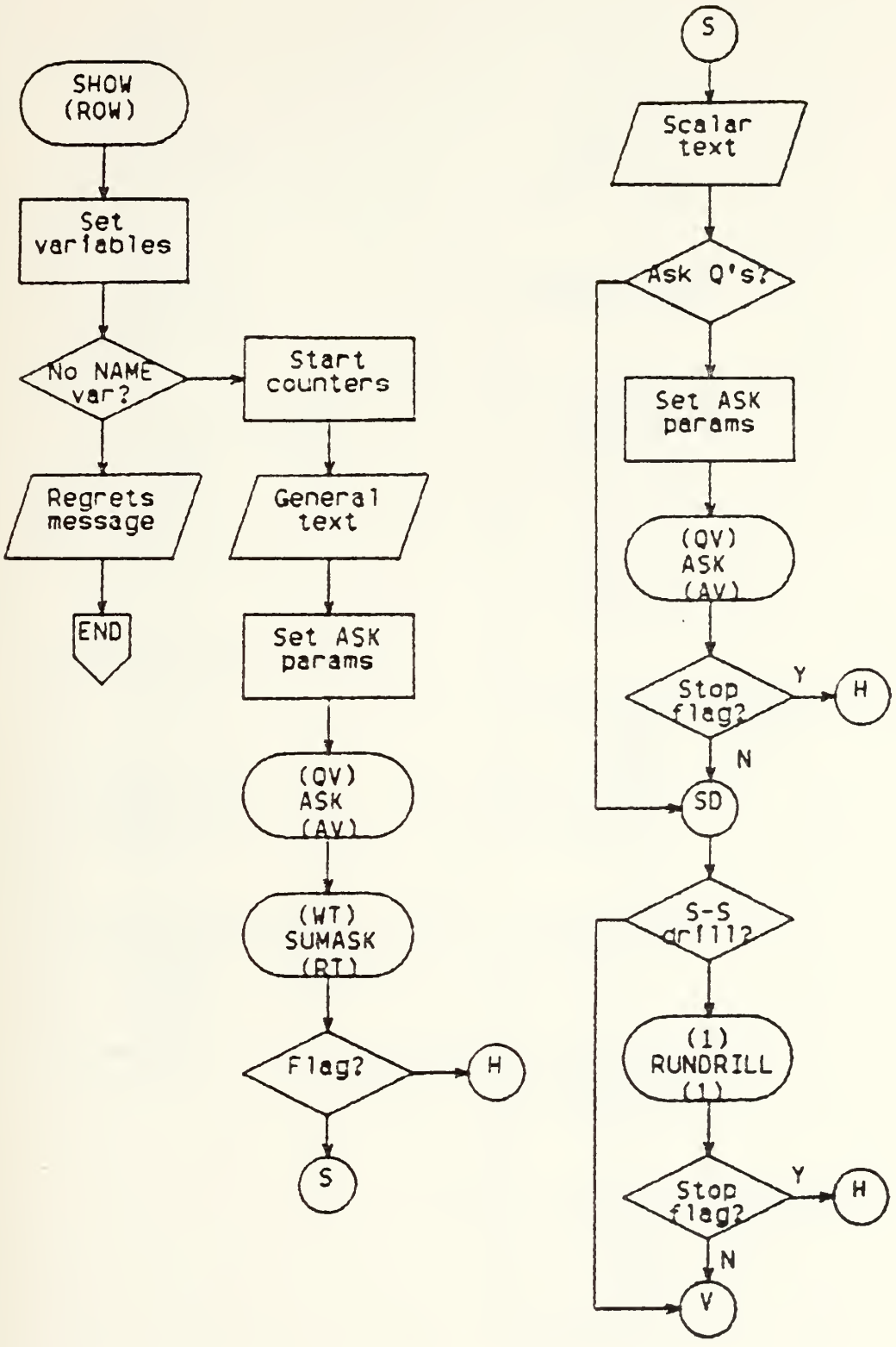

Figure B.9 Procedure of SHOW Function--1.

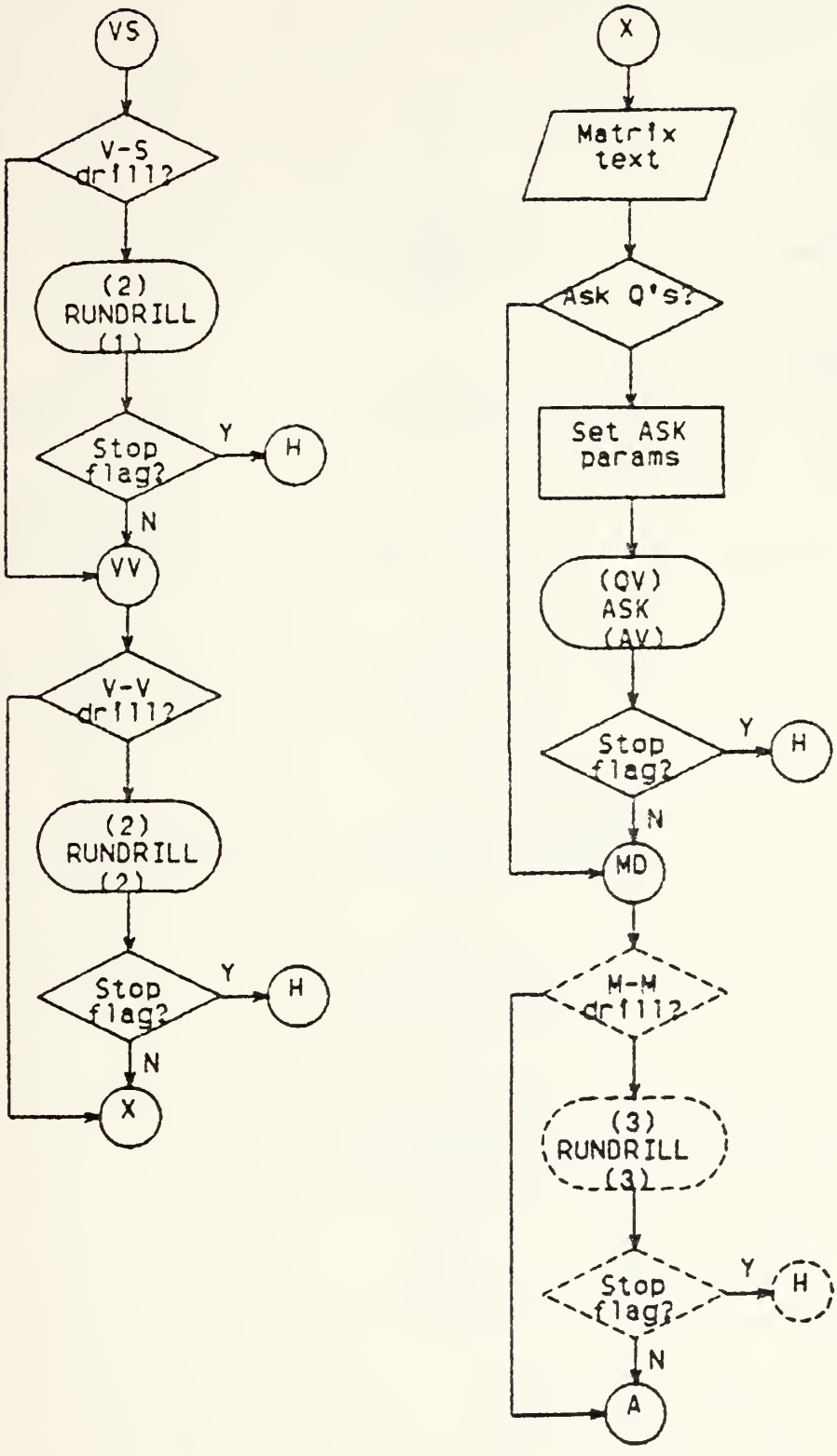

Figure B.10 Procedure of SHOW Function--2.

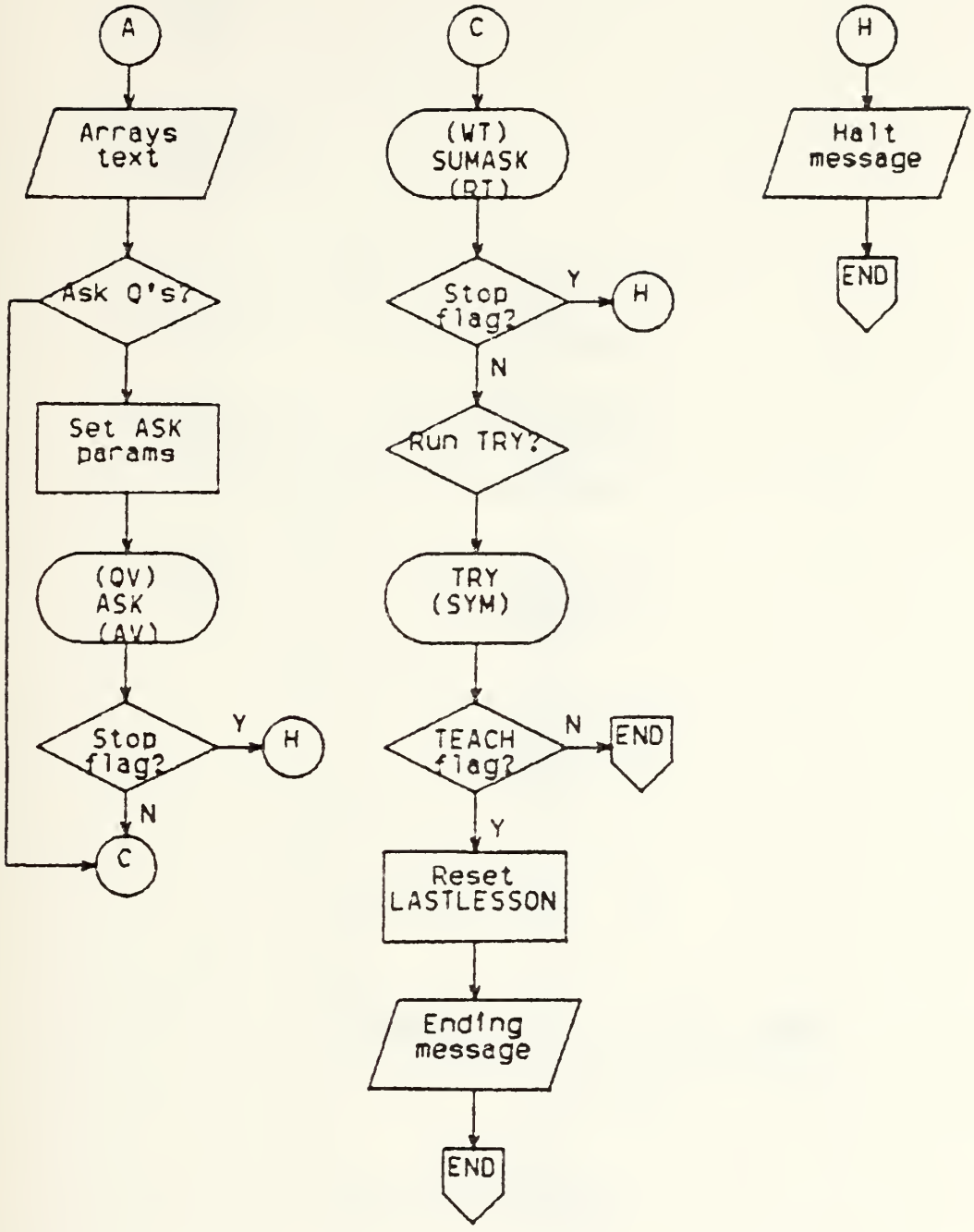

Figure B.11 Procedure of SHOW Function--3.

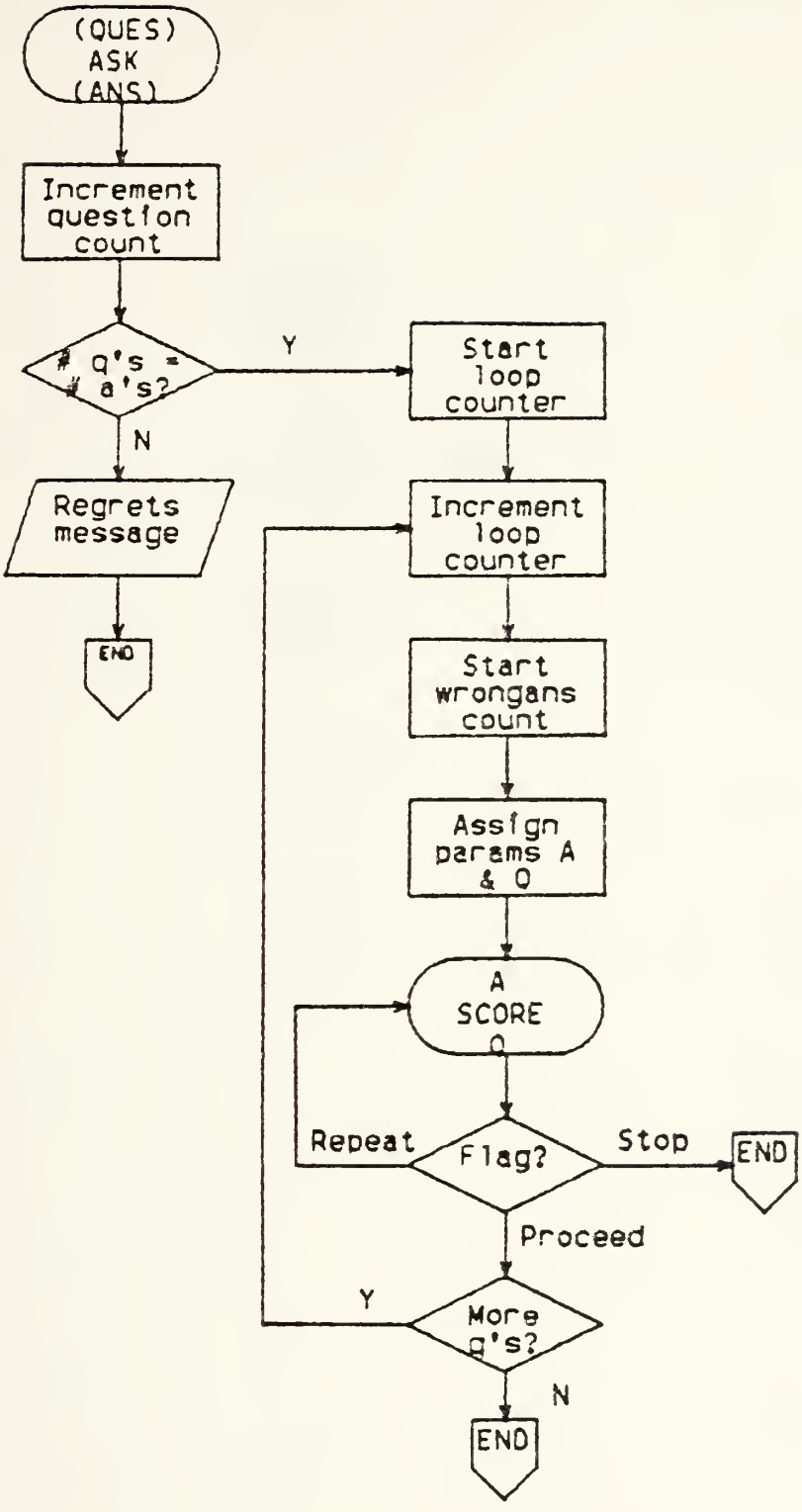

Figure B.12 Procedure of ASK Function.

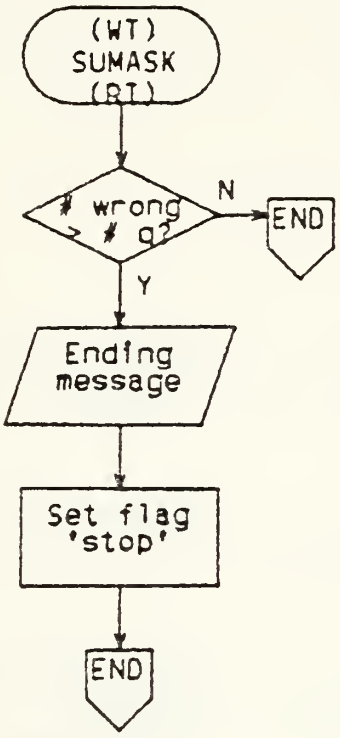

Figure B.13 Procedure of SUMASK Function.

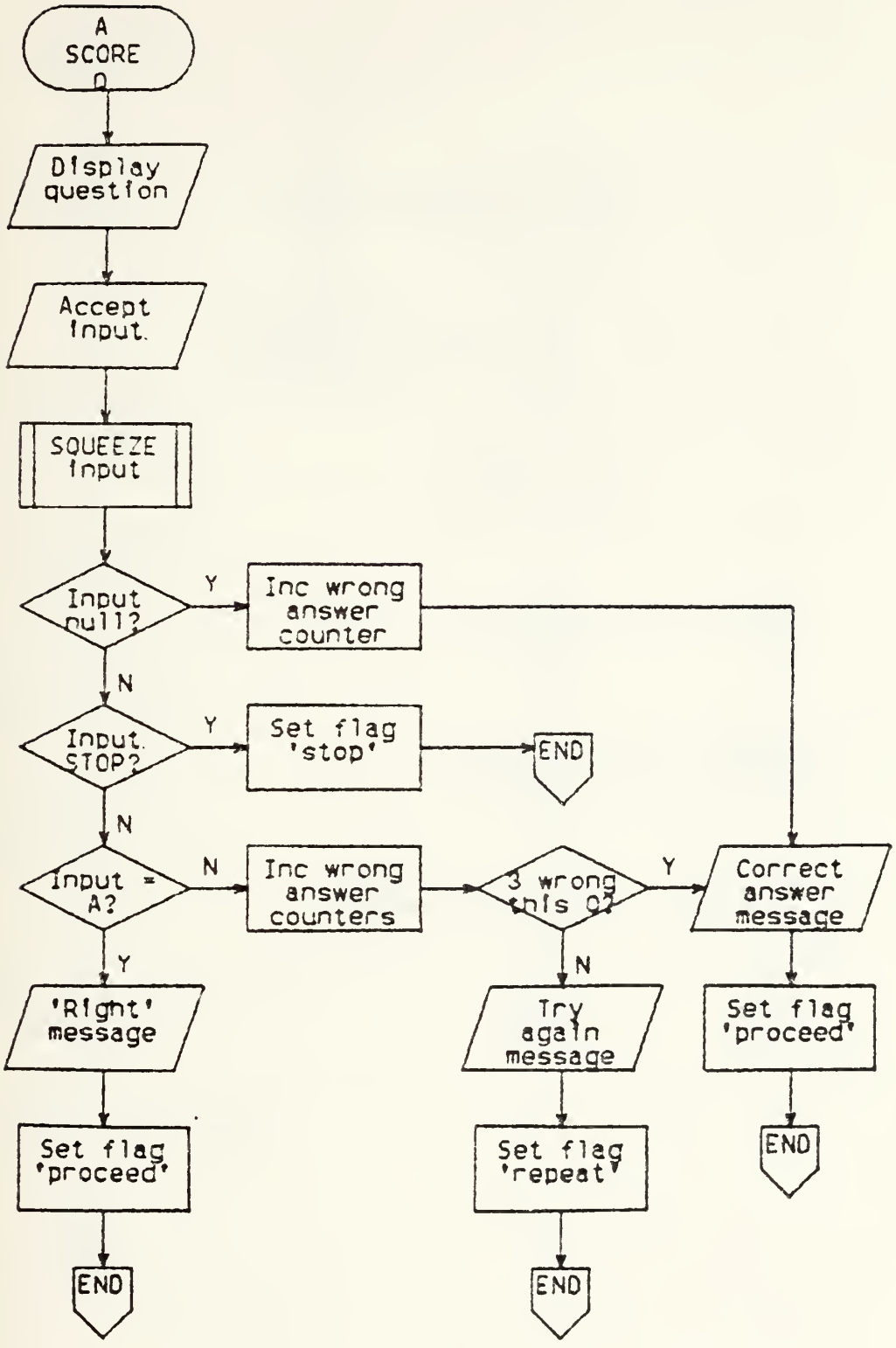

Figure B.14 Procedure of SCORE Function.

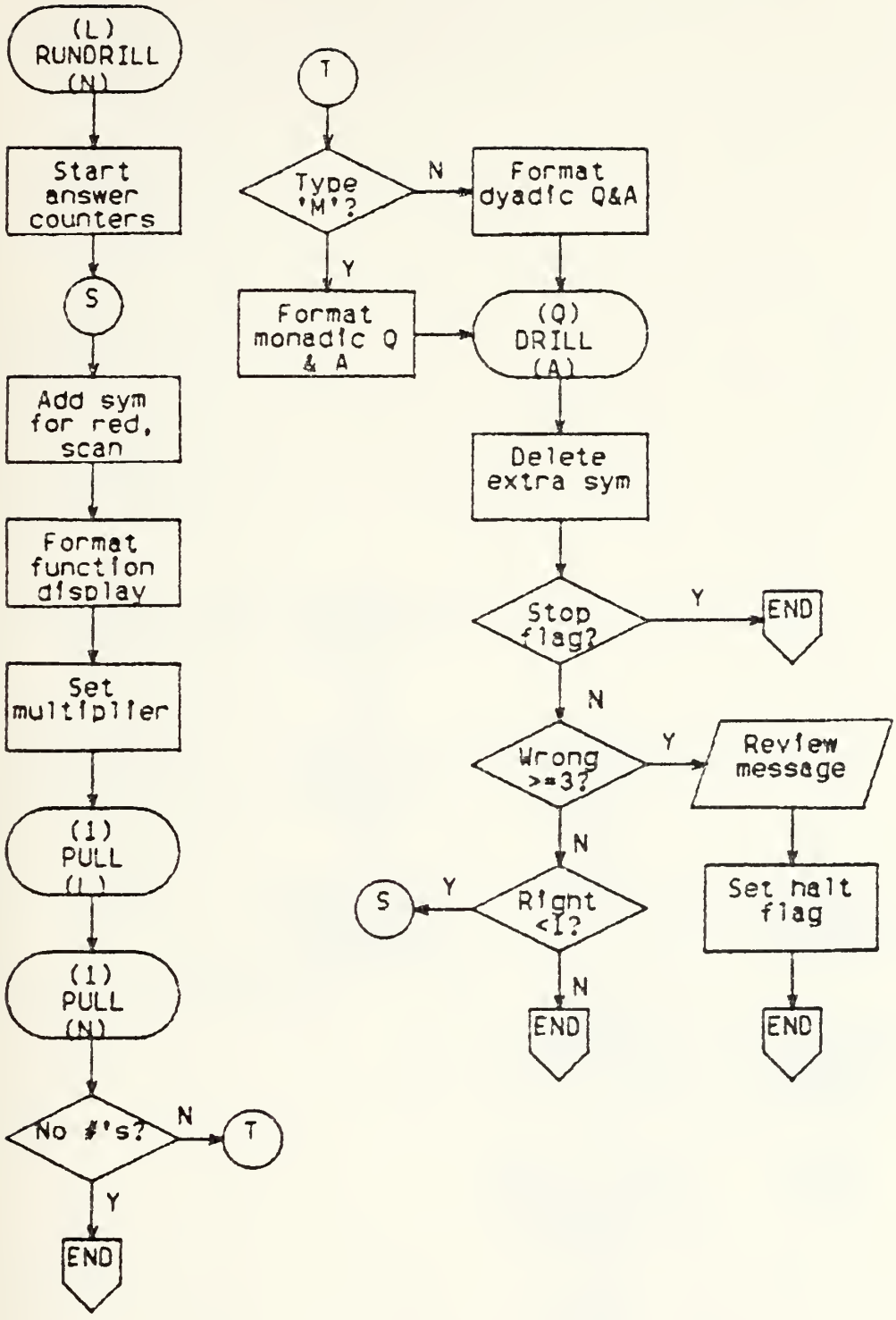

Figure B.15 Procedure of RUNDRILL Function.

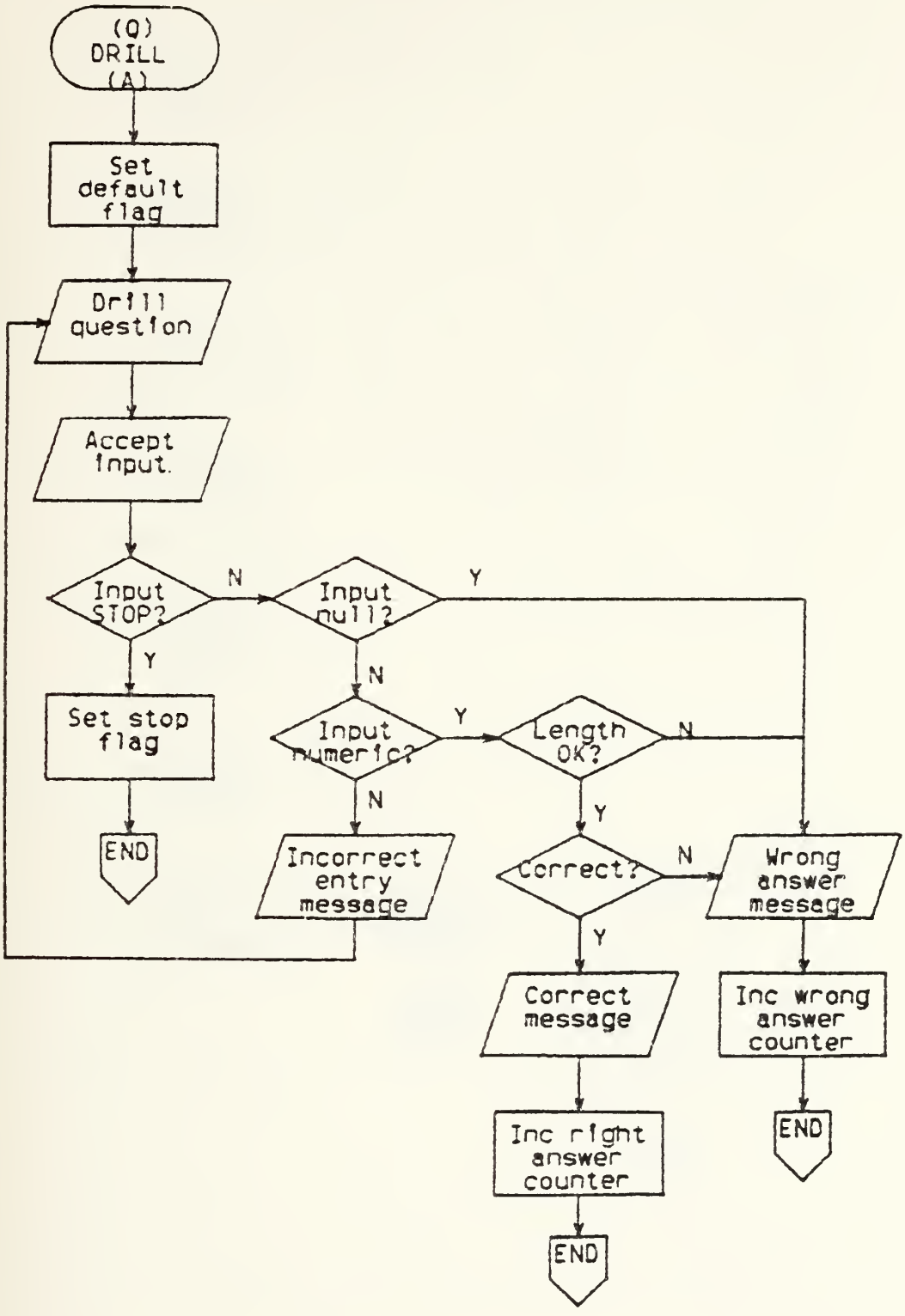

Figure B.16 Procedure of DRILL Function.

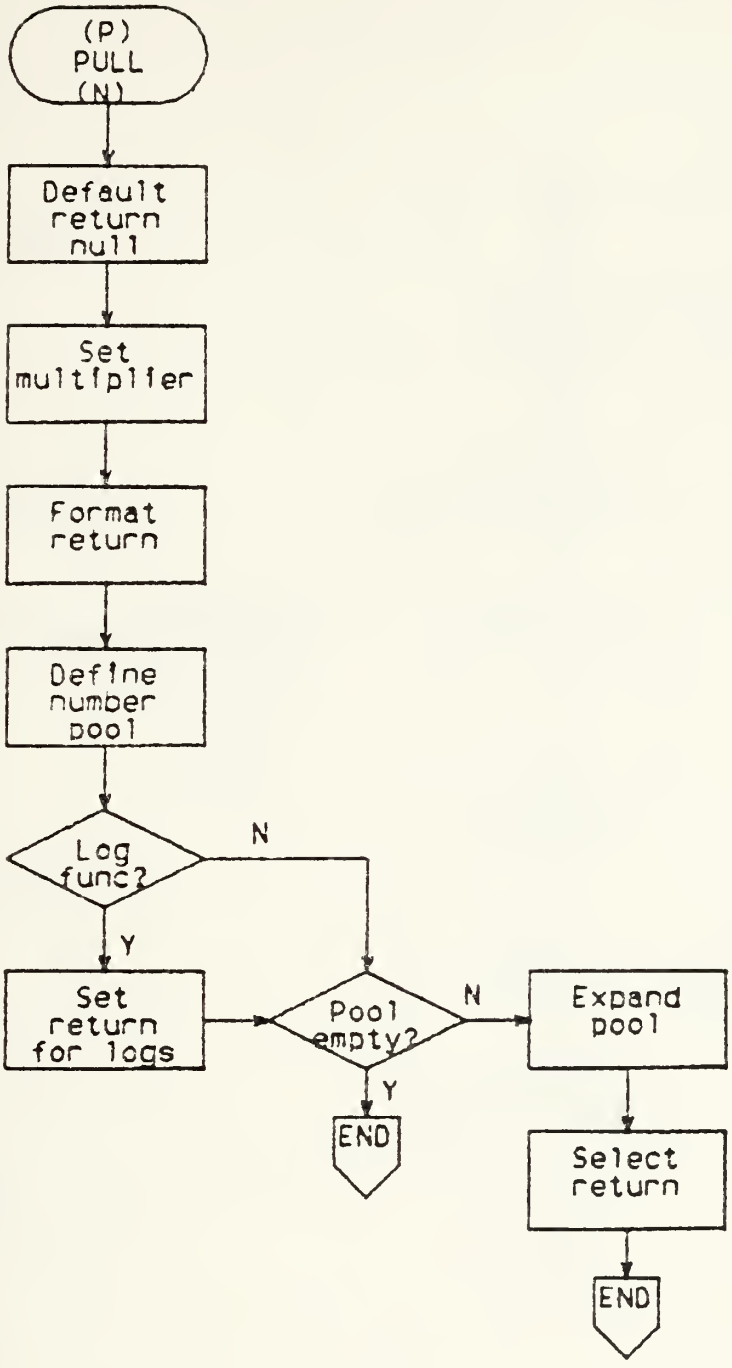

Figure B.17 Procedure of PULL Function.

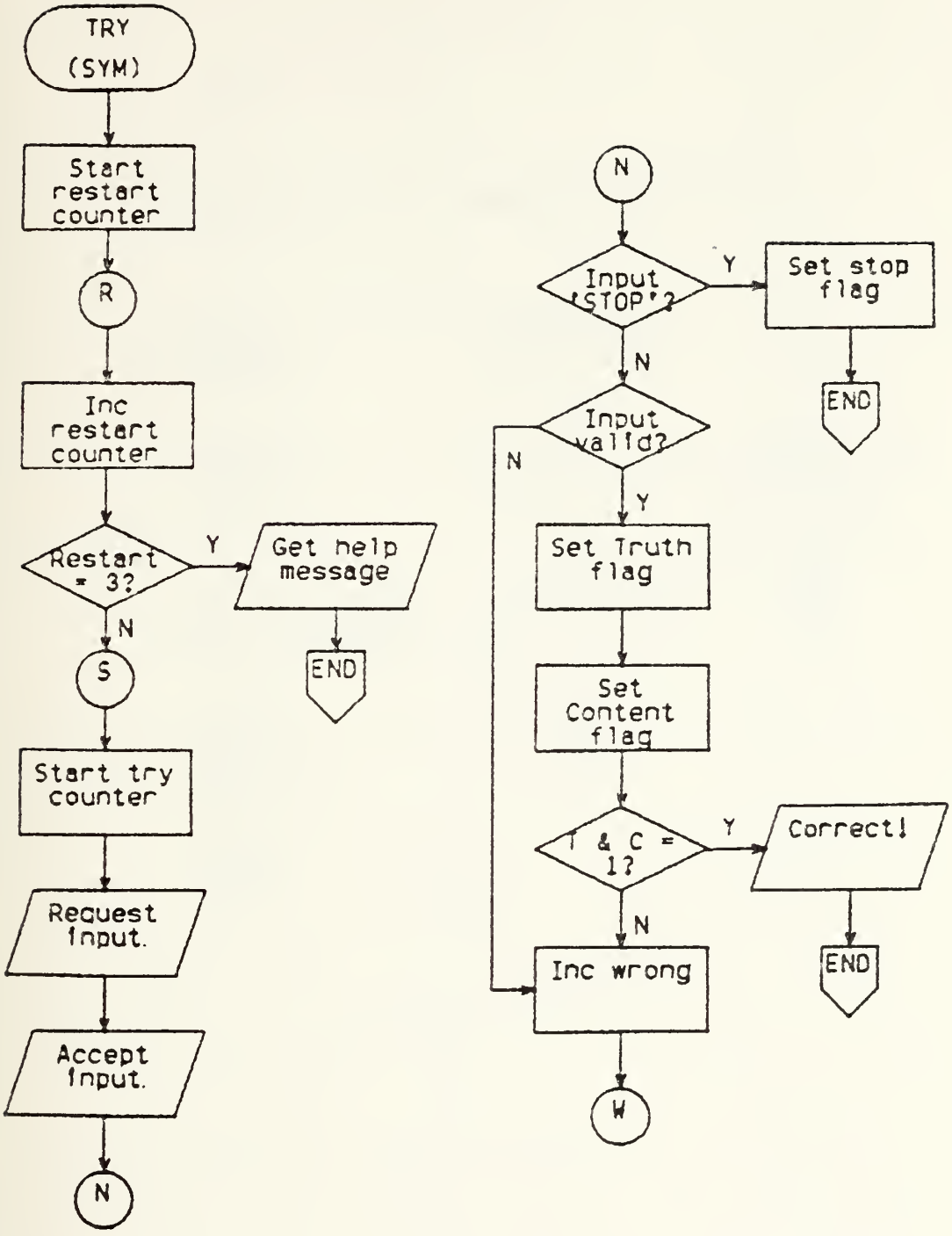

Figure B.18 Procedure of TRY Function--1.

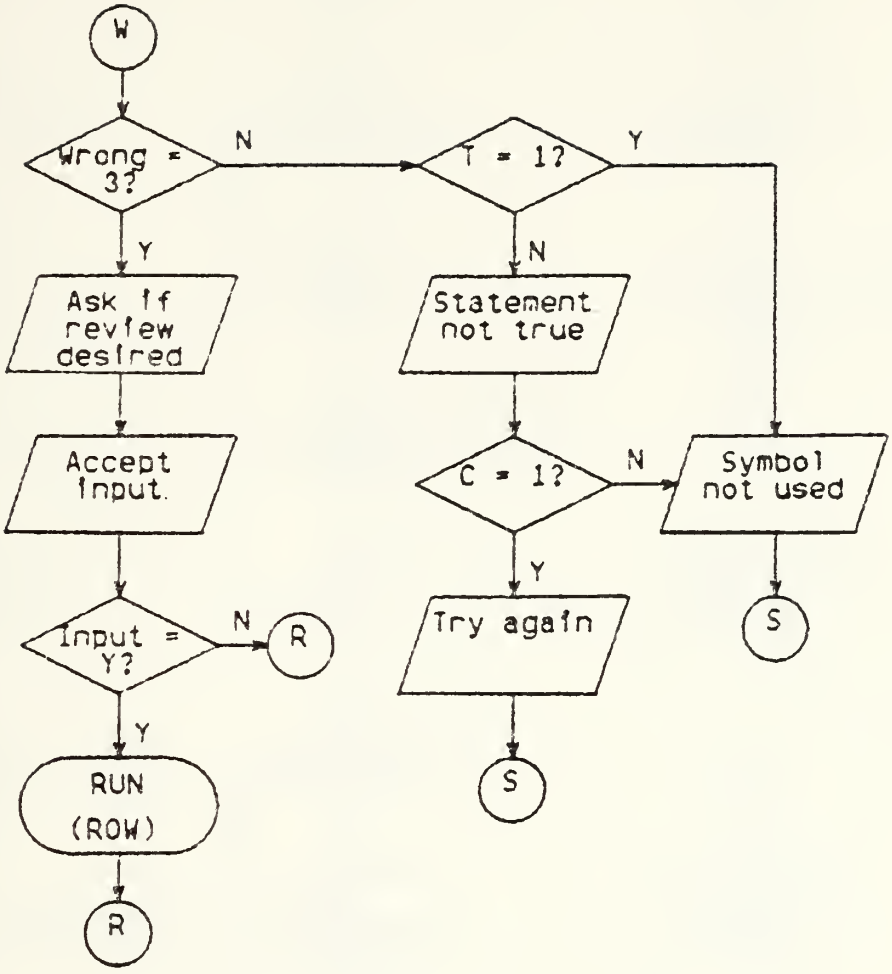

Figure B.19 Procedure of TRY Function--2.

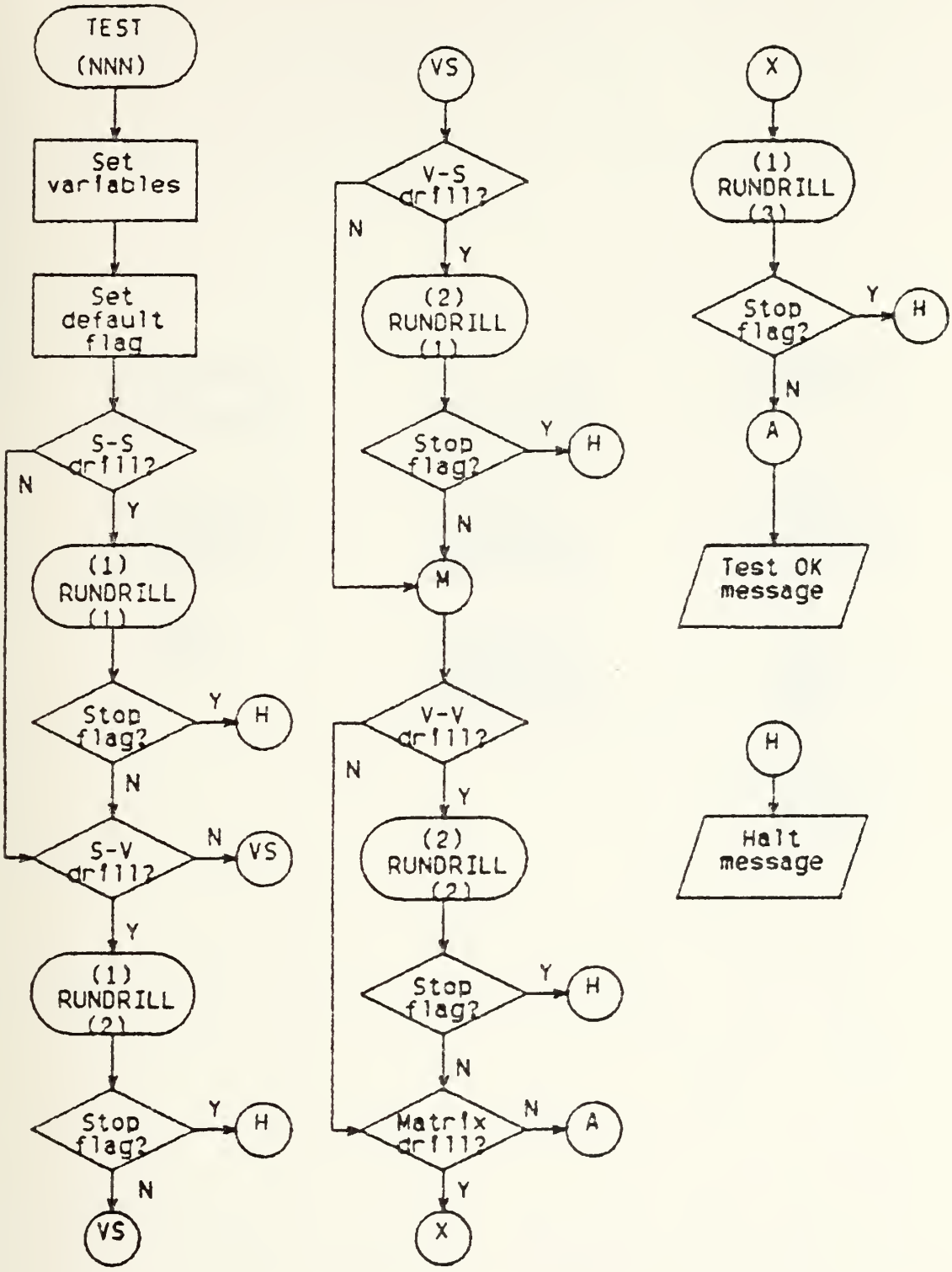

Figure B.20 Procedure of TEST Function.

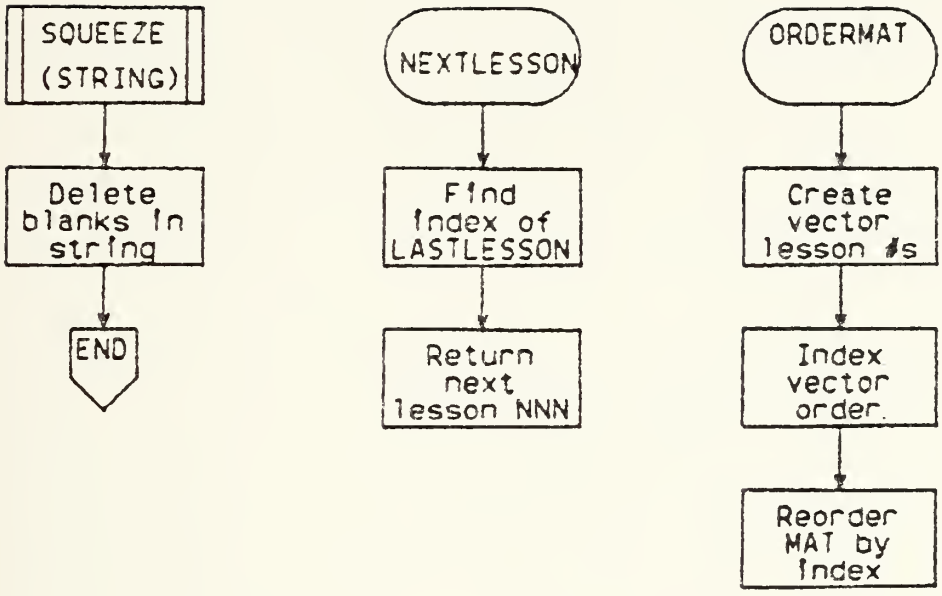

Figure B.21 Procedure of SQUEEZE, NEXTLESSON, and ORDERMAT Fns.

B. FUNCTIONS

The following section contains the APL functions which operate the TUTOR course. The comment lines, which begin with a symbol, indicate the procedure in general, but do not describe the specific programming techniques used.

```
      ∇ START
[1]   ⍝ ENSURES 'HOW' APPEARS WHEN WORKSPACE IS LOADED
[2]    ⎕LX←'HOW'
[3]   ⍝ DISPLAYS INFORMATION FOR FIRST-TIME USER
[4]    ⎕←INTRO
[5]    ⎕←BACKGROUND
[6]    ⎕←BASICS
[7]    ⎕←HOWTEACH
[8]    ⎕←⍞,'TO START YOUR FIRST LESSON IN APL, ENTER:  LESSON 101'
      ∇
      .
```

```
      ∇ MENU;ZZZ
[1]   ⍝ REQUIRES GLOBAL VARIABLE MAT
[2]    ⎕←'INFORMATION IS AVAILABLE ON THE FOLLOWING SYMBOLS/TOPICS;'
[3]   ⍝ FORMAT AND DISPLAY MENU
[4]    ZZZ←(5⍴1),0,(17⍴1),0
[5]    (((⌈((⍴MAT)[1])÷3),72)⍴ZZZ\MAT[;LESCOL,SYMCOL,FNAMECOL]
[6]    ⎕←'TO SEE MENU AGAIN, ENTER;                    MENU',CR,'FOR INFORMAT
       ION ABOUT FUNCTION, ENTER;   INFO NNN   (WHERE NNN IS MENU NUMBER)
       ',CR,'TO GO THROUGH TUTORIAL LESSON, ENTER;    LESSON NNN'
       ∇
       '
```

```
      ∇ HELP;SYMBOL;ROWS;NNN;R
[1]   ⍝ CALLS QUEUE, MENU, SQUEEZE
[2]    ⎕←'ENTER THE SYMBOL(S) YOU WOULD LIKE INFORMATION ABOUT,'
[3]    ⎕←'   OR,,,FOR MENU SELECTION, ENTER:  MENU'
[4]   ⍝ DELETE BLANKS IN RESPONSE
[5]    SYMBOL←SQUEEZE SYMBOL←⎕
[6]   ⍝ IF RESPONSE IS 'MENU', GO TO 'OTHER'; ELSE GO ON
[7]    →OTHER×⍳(4=+/+/SYMBOL∘.=⊂'MENU')
[8]   ⍝ FIND INDEXES OF ROWS THAT CONTAIN THE RESPONSE IN SYMBOL COLUMN
[9]    ROWS←(MAT[;SYMCOL]∈SYMBOL)/⍳(⍴MAT)[1]
[10]  ⍝ IF NO ROWS CONTAIN RESPONSE, GO TO 'OOPS3'; ELSE GO ON
[11]   →OOPS3×⍳0=⍴ROWS
[12]  ⍝ START LOOP COUNTER AND SET LESSON NR TO 0
[13]   NNN←R←0
[14]  ⍝ INCREMENT COUNTER
[15] NEWROW:R←R+1
[16]   ⎕←⍞
[17]  ⍝ CALL RUN TO DISPLAY HELP FUNCTION FOR ROW CORRESPONDING TO COUNTE
      R
[18]   RUN ROWS[R]
[19]  ⍝ IF MORE ROWS REMAIN, RETURN TO NEWROW; ELSE GO ON
[20]   →NEWROW×⍳R<⍴ROWS
[21]  ⍝ PRINT FUNCTION ENDING MSG AND EXIT
[22]   ⎕←'FOR MORE HELP, ENTER:   HELP'
[23]   →0
[24]  ⍝ SYMBOL NOT FOUND; PRINT ERROR MSG AND EXIT
[25] OOPS3:⎕←'SORRY, INFORMATION ABOUT THIS SYMBOL IS NOT AVAILABLE AT T
      HIS TIME,',⍞,'FOR HELP WITH ANOTHER SYMBOL, ENTER:   HELP.'
[26]   →0
[27]  ⍝ RESPONSE IS 'MENU'; EXECUTE RESPONSE AND EXIT
[28] OTHER:⍎SYMBOL
      ∇
      )
```

```
      ∇ INFO NNN;ROW
[1]   ⍝ CALLS RUN; REQUIRES GLOBAL VARIABLE MAT
[2]   ⍝ IF NNN IS NOT A SCALAR, GO TO 'OOPS1; ELSE GO ON
[3]    →OOPS1x\(⍴⍴NNN)>0
[4]   ⍝ FIND INDEXES OF ROW
[5]    ROW←((⍳⍴MAT[;LESCOL])=NNN)/\(⍴MAT)[1]
[6]   ⍝ IF NO ROWS CONTAIN RESPONSE, GO TO 'OOPS3'; ELSE GO ON
[7]    →OOPS3x\0=⍴ROW
[8]   ⍝ RUN HELP FUNCTION
[9]    RUN ROW
[10]  ⍝ DISPLAY ENDING MESSAGE
[11]   ⎕←CR,'FOR MORE INFORMATION, ENTER;  INFO NNN'
[12]   →0
[13]  ⍝ VECTOR ARGUMENT; PRINT ERROR MSG AND EXIT
[14] OOPS1:⎕←'INFO MUST BE FOLLOWED BY A SINGLE PARAMETER,',CR,'TO RESTA
      RT, ENTER;    INFO NNN     (WHERE NNN IS A THREE-DIGIT NUMBER)'
[15]   →0
[16]  ⍝ INFO NOT FOUND; PRINT ERROR MSG AND EXIT
[17] OOPS3:⎕←'SORRY, INFORMATION ABOUT THIS SYMBOL IS NOT AVAILABLE AT T
      HIS TIME.',CR,'FOR HELP WITH ANOTHER SYMBOL, ENTER;   HELP.'
      ∇
      .
```

.

```
      ∇ RUN ROW;NAME;RANK;FNAME;U
[1]   ⍝ CALLED BY HELP, INFO, TRY; CALLS SQUEEZE
[2]   ⍝ REQUIRES GLOBAL VARIABLE MAT
[3]   ⍝ASSIGN THE NAME OF THE HELP VARIABLE TO THE VARIABLE 'NAME'
[4]    NAME←,MAT[ROW;NAMECOL]
[5]   ⍝ GO TO NA IF THE HELP VARIABLE 'NAME' DOES NOT EXIST; ELSE GO ON
[6]    →NA×⍳0=⎕NC NAME
[7]   ⍝ DISPLAY THE GENERAL DESCRIPTION PORTION OF THE HELP VARIABLE
[8]    ⎕←(0 1)↓((⍕NAME)[;1]='.')⌿⍕NAME
[9]   ⍝ DISPLAY SCALAR PORTION OF HELP VARIABLE
[10]   ⎕←(0 1)↓((⍕NAME)[;1]='∘')⌿⍕NAME
[11]  ⍝ IF NO MORE INFO, EXIT; ELSE GO ON
[12]   →0×⍳~1∊(⍕NAME)[;1]∊'∨∩⍀'
[13]  ⍝ ASK IF MORE DETAILED INFO DESIRED
[14]   ⎕←⍞,'MORE?'
[15]  ⍝ IF RESPONSE IS NULL, EXIT; ELSE GO ON
[16]   →0×⍳0=⍴U←⎕
[17]   →0×⍳~'Y'=(,U)[1]
[18]  ⍝ DISPLAY VECTOR PORTION OF HELP VARIABLE
[19]   ⎕←(0 1)↓((⍕NAME)[;1]='∨')⌿⍕NAME
[20]  ⍝ DISPLAY MATRIX PORTION OF HELP VARIABLE
[21]   ⎕←(0 1)↓((⍕NAME)[;1]='∩')⌿⍕NAME
[22]  ⍝ DISPLAY ALL ARRAYS PORTION OF HELP VARIABLE
[23]   ⎕←(0 1)↓((⍕NAME)[;1]='⍀')⌿⍕NAME
[24]   →0
[25]  ⍝ ASSIGN FULL NAME OF SYMBOL TO VARIABLE 'FNAME'; PRINT ERROR MSG
[26]  NA:FNAME←SQUEEZE,MAT[ROW;FNAMECOL]
[27]   ⎕←⍞,'SORRY, INFORMATION ABOUT ',FNAME,' IS NOT AVAILABLE AT THIS
      TIME.'
      ∇
      •
```

```
     ∇ TEACH;SYMBOL;ROWS;NNN;R
[1]  ⍝ CALLS SCHEDULE, MENU, SQUEEZE
[2]   ⎕←'ENTER THE SYMBOL(S) YOU WOULD LIKE INFORMATION ABOUT,'
[3]   ⎕←'     OR FOR MENU SELECTION, ENTER:  MENU'
[4]  ⍝ DELETE BLANKS IN RESPONSE
[5]   SYMBOL←SQUEEZE SYMBOL←⎕
[6]  ⍝ IF RESPONSE IS 'MENU', GO TO 'OTHER'; ELSE GO ON
[7]   →OTHER×⍳(4=+/+/SYMBOL∘.=⊂'MENU')
[8]  ⍝ FIND INDEXES OF ROWS THAT CONTAIN THE RESPONSE IN SYMBOL COLUMN
[9]   ROWS←(MAT[;SYMCOL]∈SYMBOL)/⍳(⍴MAT)[1]
[10] ⍝ IF NO ROWS CONTAIN RESPONSE, GO TO 'OOPS3'; ELSE GO ON
[11]  →OOPS3×⍳0=⍴ROWS
[12] ⍝ INITIALIZE LOOP COUNTER AND SET LESSON NR TO 0
[13]  NNN←R←0
[14] ⍝ INCREMENT COUNTER
[15] NEWROW:R←R+1
[16]  ⎕←CR
[17] ⍝ CALL SHOW TO DISPLAY TEACH FUNCTION FOR ROW INDEXED BY COUNTER
[18]  SHOW ROWS[R]
[19] ⍝ IF ROWS REMAIN, RETURN TO NEWROW; ELSE GO ON
[20]  →NEWROW×⍳R<⍴ROWS
[21] ⍝ DISPLAY END OF SCHEDULE MSG AND EXIT
[22]  ⎕←CR,'THIS IS THE END OF SCHEDULED LESSONS.',CR,'TO SELECT MORE LE
      SSONS, ENTER:   TEACH'
[23]  →0
[24] ⍝ SYMBOL NOT FOUND; PRINT ERROR MSG AND EXIT
[25] OOPS3:⎕←'SORRY, INFORMATION ABOUT THIS SYMBOL IS NOT AVAILABLE AT T
      HIS TIME.',CR,'FOR LESSONS ON ANOTHER SYMBOL, ENTER:  TEACH'
[26]  →0
[27] ⍝ RESPONSE IS 'MENU'; EXECUTE RESPONSE AND EXIT
[28] OTHER:⍎SYMBOL
     ∇
     .
```

```
      ∇ LESSON NNN;ROW
[1]   ⍝ CALLS SHOW
[2]   ⍝ IF NNN IS NOT A SCALAR, GO TO 'OOPS1; ELSE GO ON
[3]    →OOPS1×⍳(⍴⍴NNN)>0
[4]   ⍝ SELECT ROW FOR LESSON NUMBER NNN
[5]    ROW←((⍳↑⍴MAT[;LESCOL])=NNN)/⍳(⍴MAT)[1]
[6]   ⍝ IF NO ROWS CONTAIN RESPONSE, GO TO 'OOPS3'; ELSE GO ON
[7]    →OOPS3×⍳0=⍴ROW
[8]   ⍝ CALL SHOW TO DISPLAY  TEACH FUNCTION FOR ROWS SELECTED
[9]    SHOW ROW
[10]  ⍝ DISPLAY ENDING MESSAGE AND EXIT
[11]   ⎕←'TO START ANOTHER LESSON, ENTER:   LESSON NNN'
[12]   →0
[13]  ⍝ VECTOR ARGUMENT; PRINT ERROR MSG AND EXIT
[14]  OOPS1:⎕←'LESSON MUST BE FOLLOWED BY A SINGLE PARAMETER,',CR,'TO RES
      TART, ENTER:    LESSON NNN     (WHERE NNN IS A THREE-DIGIT NUMBER)
      '
[15]   →0
[16]  ⍝ LESSON NOT FOUND; PRINT ERROR MSG AND EXIT
[17]  OOPS3:⎕←'SORRY, INFORMATION ABOUT THIS SYMBOL IS NOT AVAILABLE AT T
      HIS TIME,',CR,'FOR HELP WITH ANOTHER SYMBOL, ENTER:   HELP,'
      ∇
      )
```

```
      ∇ SHOW ROW;NAME;G;RANK;FNAME;T;RT;WT;QV;AV;TYPE;ARG;SS;F;STM
[1]   ⍝ CALLED BY SCHEDULE, LESSON; CALLS ASK, RUNDRILL, TRY, SQUEEZE
[2]   ⍝ REQUIRES GLOBAL VARIABLE MAT
[3]   ⍝ SET VARIABLES CORRESPONDING TO ROW
[4]    NAME←,MAT[ROW;NAMECOL]
[5]    FNAME←SQUEEZE,MAT[ROW;FNAMECOL]
[6]    STM←MAT[ROW;STMCOL]
[7]    RANK←MAT[ROW;SVXACOL]
[8]    TYPE←MAT[ROW;MDCOL]
[9]    ARG←MAT[ROW;NKCOL]
[10]  ⍝ GO TO 'NA' IF HELP VARIABLE DOES NOT EXIST; ELSE GO ON
[11]   →NA×⍳(0=⎕NC NAME)
[12]  ⍝ INITIALIZE TOTAL QUESTION, WRONG ANSWER COUNTERS
[13]   RT←WT←0
[14]   T←⍳0
[15]  ⍝-----------------------------------------------------------------
[16]  ⍝ DISPLAY GENERAL PORTION OF HELP VARIABLE
[17]   ⎕←(0 1)↓((↕NAME)[;1]='.')⌿↕NAME
[18]  ⍝ GENERAL QUESTIONS
[19]   QV←1,(5×⍳TYPE='D'),6
[20]   AV←(↑1+TYPE='D'),SQUEEZE RANK
[21]   QV ASK AV
[22]  ⍝ CHECK NUMBER OF RIGHT ANSWERS
[23]   WT SUMASK RT
[24]  ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[25]   →HALT×T=2
[26]  ⍝-----------------------------------------------------------------
[27]  ⍝ DISPLAY SCALAR PORTION OF HELP VARIABLE
[28]   ⎕←(0 1)↓((↕NAME)[;1]='∘')⌿↕NAME
[29]  ⍝ SCALAR QUESTIONS
[30]   →SD×⍳TYPE='M'
[31]   QV←2
[32]   AV←'NT'[1+STM∈'+×⌈⌊≠∨∗∧⍲']
```

```
[33]  ⍞V ASK AV
[34] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[35]  →HALTXY=2
[36] ⍝ SCALAR DRILL
[37] SD:→VX⍳⍵SYM⍷SS←'+-X÷⌈⌊!|⍵⍳⍟(⍳=≥)≠∨∧⍲⍱,⍴⍕'
[38]  →VX⍳((SYM⍷'+⍟⍴,')∧TYPE='M')∨(SYM⍷'⍳')∧TYPE='D'
[39]  1 RUNDRILL 1
[40] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[41]  →HALTXY=2
[42] ⍝------------------------------------------------------------
[43] ⍝ DISPLAY VECTOR PORTION OF HELP VARIABLE
[44] V:⎕←(0 1)↓((⍎NAME)[;1]='∨')⌿⍎NAME
[45] ⍝ ASK VECTOR QUESTIONS
[46]  →VDX⍳TYPE='M'
[47]  ⍞V←3
[48]  AV←'FT'[1+SYM⍷SS,'←[⍷⌽⍉⍋↑↓⊥⊤+']
[49]  ⍞V ASK AV
[50] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[51]  →HALTXY=2
[52] ⍝ VECTOR DRILL
[53] VD:→XX⍳(⍵SYM⍷SS,'⍒⍋⍷/\⌿⍀↑↓⌽⍉')∨((SYM⍷',+⍟')∧TYPE='M')∨(SYM⍷'/\⌿⍀')∧
      TYPE='D'
[54]  →VSX⍳(SYM⍷'⍳')
[55]  1 RUNDRILL 2
[56] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[57]  →HALTXY=2
[58] VS:→VVX⍳(SYM⍷'⌽⍴⍒⍋')∧TYPE='M'
[59]  →XX⍳(SYM⍷'⍴↑↓⌽⍉')∨TYPE='M'
[60]  2 RUNDRILL 1
[61] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[62]  →HALTXY=2
[63] VV:→XX⍳SYM⍷'⍳'
[64]  2 RUNDRILL 2
[65] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
```

```
[66]  →HALTXY=2
[67] ⍝-----------------------------------------------------------------
[68] ⍝ DISPLAY MATRIX PORTION OF HELP VARIABLE
[69] X;⎕←(0 1)↓((⍷NAME)[;1]='⍝')/⍷NAME
[70] ⍝ ASK MATRIX QUESTIONS
[71]  →AX\TYPE='M'
[72]  QV←4
[73]  AV←'FT'[1+SYM∈'+-×÷*⍟|⌈⌊=≠<≤≥>∨⍱∧⍲']
[74]  QV ASK AV
[75] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[76]  →HALTXY=2
[77] ⍝ MATRIX DRILL
[78] MD; ⍝TYPE RUNDRILL 3
[79] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[80]  →HALTXY=2
[81] ⍝-----------------------------------------------------------------
[82] ⍝ DISPLAY ALL ARRAYS PORTION OF HELP VARIABLE
[83] A;⎕←(0 1)↓((⍷NAME)[;1]='⍵')/⍷NAME
[84] ⍝ ASK ALL ARRAYS QUESTIONS
[85]  QV←8↑9↑X\XF←+/(1 2)×,ARG∈'KE'
[86]  AV←('NY'[1+XF]),SQUEEZE ' LRB'[1+F]
[87]  QV ASK AV
[88] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[89]  →HALTXY=2
[90] ⍝-----------------------------------------------------------------
[91] ⍝ SUMMARIZE STUDENT PERFORMANCE
[92] SUM;WT SUMASK RT
[93] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[94]  →HALTXY=2
[95] ⍝-----------------------------------------------------------------
[96] ⍝ CALL TRY FOR SELECTED SYMBOLS
[97] EXP;→ENDX\∼(SYM∈'+-×÷')∧TYPE='D'
[98]  TRY SYM
[99] ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
```

```
[100]  →HALTxT=2
[101] ⍝-----------------------------------------------------------------
[102] ⍝ LESSON COMPLETE; IF NNN=0 (TEACH), EXIT; ELSE GO ON
[103] END:→0x\NNN=0
[104] ⍝ NNN≠0 (LESSON); SET LASTLESSON TO THIS LESSON NR
[105] OUT:LASTLESSON←NNN
[106] ⍝ DISPLAY MSG AND EXIT
[107]  ⎕←CR,'THIS IS THE END OF YOUR LESSON,'
[108]  ⎕←'FOR THE NEXT LESSON IN SEQUENCE, ENTER:   LESSON NEXTLESSON'
[109]  →0
[110] ⍝ HELP VAR NOT AVAILABLE; DISPLAY ERROR MSG AND EXIT
[111] NA:CR,'SORRY, INFORMATION ABOUT ',FNAME,' IS NOT AVAILABLE AT THIS
        TIME,'
[112]  →0
[113] HALT:CR,'THIS LESSON HAS BEEN HALTED,'
      ∇
      '
```

```
      ∇ QUES ASK ANS;R;W;I;A;L;AR;RK;Q
[1]   ⍝ CALLED BY SHOW; CALLS SCORE
[2]   ⍝ REQUIRES NON-LOCAL VARIABLES RT,WT,Y, GLOBAL CUES
[3]   ⍝ ASSIGN THE NUMBER OF QUESTIONS TO 'R'
[4]    R←⍴,QUES
[5]   ⍝ ASSIGNS TOTAL NUMBER OF QUESTIONS TO 'RT'
[6]    RT←RT+R
[7]   ⍝ IF NR OF QUESTIONS ≠ NR OF ANSWERS, GO TO 'OOPS'; ELSE GO ON
[8]    →OOPS×⍳R≠⍴,ANS
[9]   ⍝ INITIALIZE COUNTER, CHARACTER VARS
[10]   I←0
[11]  ⍝ START LOOP; INCREMENT COUNTER; INITALIZE WRONG ANSWER COUNT
[12]  START:I←I+1
[13]   W←0
[14]  ⍝ ASSIGN QUESTION AND ANSWER PARAMETERS FOR ASK FUNCTION
[15]   A←(,ANS)[I]
[16]   Q←⊆CR,⍕CUES[(,QUES)[I];]
[17]  ⍝ CALL 'SCORE' TO EVALUATE THE QUESTION
[18]  QUERY:A SCORE Q
[19]  ⍝ IF FLAG INDICATES 'REPEAT', RETURN TO SAME QUESTION;
[20]  ⍝ IF FLAG INDICATES 'STOP' RESPONSE, EXIT; ELSE GO ON
[21]   →(QUERY,0)[Y]
[22]  ⍝ IF QUESTIONS REMAIN, RETURN TO 'START'; ELSE EXIT
[23]   →START×⍳I<R
[24]   →0
[25]  ⍝ FOR DEBUGGING; ARGUMENTS NOT SAME LENGTH; PRINT ERROR MSG AND EXI
      T
[26]  OOPS:⎕←'⍴QUES ≠ ⍴ANS'
      ∇
      •
```

```
      ∇ A SCORE Q;INPUT
[1]   ⍝ CALLED BY ASK
[2]   ⍝ REQUIRES NON-LOCAL VARIABLES Y,W,WT
[3]   ⍝ DISPLAY TEXT OF QUESTION
[4]    ⎕←Q
[5]   ⍝ ASSIGN RESPONSE TO 'INPUT' AFTER DELETING BLANKS
[6]    INPUT←SQUEEZE INPUT←⍞
[7]   ⍝ IF RESPONSE IS NULL, ADD 1 WRONG ANS AND GO TO 'LAST'; ELSE GO ON
[8]    WT←WT+0=⍴INPUT
[9]    →LASTx⍳0=⍴INPUT
[10]  ⍝ IF RESPONSE IS 'STOP', GO TO 'OUT'; ELSE GO ON
[11]   →OUTx⍳4=+/+/INPUT∘.='STOP'
[12]  ⍝ IF RESPONSE ≠ RIGHT ANSWER, GO TO 'WRONG'; ELSE GO ON
[13]  NEXT:→WRONGx⍳(,INPUT)[1]≠A
[14]  ⍝ DISPLAY MSG
[15]   ⎕←'RIGHT'
[16]  ⍝ SET FLAG TO NULL; EXIT
[17]   →0,Y←⍳0
[18]  ⍝ INCREMENT WRONG ANSWER COUNTERS
[19]  WRONG:W←W+1
[20]   WT←WT+1
[21]  ⍝ GO TO LAST IF THREE WRONG ANSWERS TO THIS QUESTION; ELSE GO ON
[22]   →LASTx⍳W≥3
[23]  ⍝ DISPLAY MSG
[24]   ⎕←'SORRY, PLEASE TRY AGAIN'
[25]  ⍝ SET FLAG TO INDICATE REPEAT QUESTION; EXIT
[26]   →0,Y←1
[27]  ⍝ →0
[28]  ⍝ THREE WRONG ANSWERS; DISPLAY MSG
[29]  LAST:⎕←'SORRY, THE CORRECT ANSWER IS ',A
[30]  ⍝ SET FLAG TO INDICATE GO TO NEXT QUESTION; EXIT
[31]   →0,Y←⍳0
[32]  ⍝ SET FLAG TO INDICATE 'STOP' RESPONSE
[33]  OUT:Y←2
      ∇
```

```
      ∇ WT SUMASK R
[1]   ⍝ CALLED BY SHOW AFTER ALL 'ASK' CALLS
[2]   ⍝ IF FEWER WRONG RESPONSES THAN QUESTIONS, EXIT; ELSE GO ON
[3]    →0×⍳WT≤R
[4]   ⍝ WRONG RESPONSES ≥ QUESTIONS; DISPLAY MESSAGE AND EXIT
[5]    ⎕←'YOU ENTERED ',(⍕WT),' WRONG ANSWERS OUT OF ',(⍕R),' QUESTIONS.'
[6]    ⎕←'IF YOU WANT TO RETAKE THIS LESSON, ENTER;   LESSON ',⍕,MAT[ROW;
       LESCOL]
[7]    Y←2
       ∇
       .
```

```
      ∇ L RUNDRILL N;FUNC;R;K1;K2;Q;A;W;C;M
[1]   ⍝ CALLED BY SHOW; CALLS DRILL,PULL
[2]   ⍝ REQUIRES NON-LOCAL VARIABLES SYM, TYPE
[3]   ⍝ INITIALIZE ANSWER COUNTERS
[4]    C←W←0
[5]   START:
[6]   ⍝ EXTEND SYM FOR REDUCTION, SCAN
[7]    SYM←((SYM∈'/\⌿⍀')/('+-⌈⌊=')[?5]),SYM
[8]   ⍝ FORMAT FUNCTION DISPLAY
[9]    R←(1,1,2)[N]
[10]   FUNC←(R,3)⍴' ',SYM,'     '
[11]  ⍝ SET MULTIPLIER
[12]   M←?3
[13]  ⍝ PULL RANDOM NUMBER ARGUMENTS
[14]   K1←1 PULL L
[15]   K2←2 PULL M
[16]  ⍝ IF K1 IS EMPTY VECTOR, EXIT; ELSE GO ON
[17]   →0×⍳0=⍴,K1
[18]  ⍝ IF TYPE IS MONADIC, GO TO MONTY; ELSE GO ON
[19]   →MONTY×⍳TYPE='M'
[20]  ⍝ FORMAT DYADIC Q / A
[21]   Q←(↑K1),FUNC,↑K2
[22]   A←⍕(↑,K1),SYM,↑,K2
[23]   →RUN
[24]  ⍝ FORMAT MONADIC Q/A
[25]  MONTY:
[26]   Q←FUNC,↑K2
[27]   A←⍕SYM,↑,K2
[28]  ⍝ RUN DRILL
[29]  RUN:Q DRILL A
[30]  ⍝ DELETE EXTRA SYMBOL (IF ANY)
[31]   SYM←¯1↑SYM
[32]  ⍝ IF STOP FLAG IS SET, EXIT; ELSE GO ON
```

```
[33]  →0×T=2
[34] ⍝ IF WRONG ANSWERS ≥ 3, GO TO 'REV'; ELSE GO ON
[35]  →REV×⍳W≥3
[36] ⍝ IF RIGHT ANSWERS < 1, RETURN TO 'START'; ELSE EXIT
[37]  →START×⍳C<1
[38]  →0
[39] REV:
[40]  ⎕←CR,'PLEASE REVIEW THE DESCRIPTION OF THIS FUNCTION OR TALK WITH
      YOUR',CR,'   INSTRUCTOR BEFORE RETURNING TO THIS UNIT.'
[41]  T←2
      ∇
      .
```

```
      ∇ Z←P FULL N;R;C;L;H;E;POOL
[1]   ⍝ CALLED BY RUNDRILL
[2]   ⍝ REQUIRES NON-LOCAL VARIABLES STM, M
[3]   ⍝ SET DEFAULT VALUE
[4]    Z←⍳0
[5]   ⍝ P IS 1 FOR LEFT ARG, 2 FOR RIGHT ARG
[6]   ⍝ SET MULTIPLIER BASED ON M FROM 1 TO 3(EXCEPT MONADIC ⌈L)
[7]    MULT←((10*M-2),0.1)[1+(STM[1]∈'⌈L')∧(TYPE='M')∧∨(¯1↑STM)∈'/⍀⌿']
[8]   ⍝ N IS RANK OF DESIRED RANDOM ARRAY; ASSIGN ROWS AND COLUMNS TO R A
      ND C
[9]    R←(1,1,2)[N]
[10]   C←(1,(1+M),2)[N]
[11]  ⍝ DETERMINE LIMITS BASED ON TYPE OF SYMBOL
[12] POND:POOL←⍳0
[13]   POOL←POOL,(STM[1]∈'+-×⌈L,<≤=≥>≠∈⍷⍸')/MULT×H,0,-H←⍳9
[14]   POOL←POOL,((STM[1]∈'↑↓⍴⌽⍉⊖')∧P=2)/MULT×H,0,-H
[15]   POOL←POOL,((STM[1]∈'↑↓⌽⍉⊖')∧P=1)/E,-E←⍳1⌈M
[16]   POOL←POOL,((STM[1]∈'⌹!⍳*⍴'))/2,3,4
[17]   POOL←POOL,(STM[1]∈'÷')/MULT×D,-D←2,4,5,8
[18]   POOL←POOL,(STM[1]∈'∨∧⍱⍲∼')/ 0 1
[19]   POOL←POOL,((STM[1]='●')∧TYPE='M')/1,*⍳3
[20]   →PZ×⍳(TYPE='M')∨STM[1]≠'●'
[21]  ⍝ KLUGE FOR LOGARITHM--K2 ALWAYS A POWER OF K1 (POOL IS EMPTY)
[22]   POOL←POOL,((STM[1]='●')∧(TYPE='D')∧P=1)/2,3,10
[23]   →PZ×⍳P=1
[24]   Z←(R,C)⍴((STM[1]='●')∧TYPE='D')/(,K1)*?(⍴,K1)⍴3
[25]  ⍝ IF NO NUMBERS IN POOL, EXIT; ELSE GO ON
[26] PZ:→0×⍳0=⍴POOL
[27]   POOL←100⍴POOL
[28]  ⍝ PROVIDE ARRAY OF RANDOM NRS
[29]   Z←(R,C)⍴POOL[(R×C)?100]
      ∇
      •
```

```
      ∇ Q DRILL A;B;SB;SQ;L
[1]   ⍝ CALLED BY RUNDRILL; CALLS SQUEEZE
[2]   ⍝ REQUIRES NON-LOCAL VARIABLES C,W; GLOBAL CR
[3]   START:
[4]    T←⍳0
[5]   ⍝ DISPLAY DRILL QUESTION
[6]   ASK:⎕←CR,'WHAT IS THE RESULT OF:'
[7]    ⎕←Q
[8]   ⍝ IF REPONSE IS 'STOP', GO TO 'HALT'; ELSE GO ON
[9]    →HALTX⍳4=+/+/'STOP'∘.=SB←SQUEEZE B←⎕
[10]  ⍝ IF ANSWER IS NULL, GO TO WR; ELSE GO ON
[11]   →WRX⍳0=⍴SB
[12]  ⍝ CHECK FOR NUMERIC RESPONSE
[13]  CK:→NAX⍳(0∈SB∈'1234567890.¯')∨∧/0∈SB∈'¯.'
[14]  ⍝ IF LENGTH OF RESPONSE ≠ LENGTH OF GIVEN ANSWER, GO TO 'WR'; ELSE
      GO ON
[15]   →WRX⍳(⍴,A)≠⍴,⍎B
[16]  ⍝ IF RESPONSE = GIVEN ANSWER, GO TO 'RT'; ELSE GO ON
[17]   →RTX⍳((,A)∧.=,⍎B)
[18]  ⍝ ANSWER IS WRONG; DISPLAY ANSWER
[19]  WR:⎕←'SORRY, CORRECT ANSWER IS:'
[20]   ⎕←A
[21]  ⍝ INCREMENT WRONG ANSWER COUNTER AND RETURN TO RUNDRILL
[22]   W←W+1
[23]   →0
[24]  ⍝ ANSWER IS RIGHT; DISPLAY MESSAGE
[25]  RT:⎕←'CORRECT!'
[26]  ⍝ INCREMENT RIGHT ANSWER COUNTER AND RETURN
[27]   C←C+1
[28]   →0
[29]  NA:⎕←'SORRY, THIS ANSWER NOT ACCEPTABLE, PLEASE ENTER A NUMERIC ANS
      WER,'
[30]   →ASK
```

```
[31] ⍝ STOP AT STUDENT REQUEST; SET FLAG AND EXIT
[32] HALT:→0,Y←2
     ∇
     .
```

```
      ∇ TRY STM;W;B;T;C;TW
[1]   ⍝ CALLED BY SHOW; CALLS RUN
[2]   ⍝ REQUIRES NON-LOCAL VARIABLES ROW, STM, FNAME
[3]    TW←0
[4]   RESTART:TW←TW+1
[5]   ⍝ GO TO OUCH IF RESTART COUNTER EQUALS 3
[6]    →OUCH×⍳TW=3
[7]   ⍝ INITIALIZE FLAGS AND WRONG ANSWER COUNTER
[8]    T←C←W←0
[9]   ⍝ REQUEST STUDENT INPUT A LOGICAL EXPRESSION USING THE SYMBOL
[10]  START:
[11]   ⎕←CR,'WRITE A TRUE STATEMENT USING ',STM,',',CR,'FOR EXAMPLE: 4=2+
       2'
[12]  ⍝ IF RESPONSE IS 'STOP', GO TO 'HALT'; ELSE GO ON
[13]   →HALT×⍳4=+/+/'STOP'∘.=B←⍞
[14]  ⍝ IF RESPONSE IS A NULL OR A NAME, GO TO 'WRONG'; ELSE GO ON
[15]   →WRONG×⍳(4≠⎕NC B)∨0=⍴B
[16]  ⍝ SET T=1 IF STATEMENT IS TRUE
[17]  NEXT:T←1=⍎B
[18]  ⍝ SET C=1 IF STATEMENT CONTAINS THE SYMBOL
[19]   C←STM∈B
[20]  ⍝ GO TO RIGHT IF T AND C BOTH EQUAL 1; ELSE GO ON
[21]   →RIGHT×⍳T∧C
[22]  ⍝ ANSWER IS WRONG; INCREMENT COUNTER
[23]  WRONG:W←W+1
[24]  ⍝ GO TO REV IF WRONG ANSWER COUNTER EQUALS 3
[25]   →REV×⍳W=3
[26]  ⍝ GO TO NOSTM IF THE STATEMENT IS TRUE; ELSE GO ON
[27]   →NOSTM×⍳T
[28]  ⍝ PRINT ERROR MSG
[29]  NT:⎕←'SORRY, THIS STATEMENT IS NOT TRUE,'
[30]  ⍝ GO TO NOSTM IF THE STATEMENT DOES NOT CONTAIN THE SYMBOL; ELSE GO
       ON
```

```
[31]  →NOSYMx\~C
[32] ⍝ PRINT MSG AND RETURN TO START
[33]  ⎕←'TRY AGAIN,'
[34]  →START
[35] ⍝ PRINT ERROR MSG AND RETURN TO START
[36] NOSYM:⎕←'YOUR ANSWER DOES NOT USE ',SYM,', TRY AGAIN,'
[37]  →START
[38] ⍝ PRINT MSG AND EXIT
[39] RIGHT:⎕←'CORRECT!'
[40]  →0
[41] ⍝ PRINT MSG AND RESTART
[42] REV:⎕←⍞,'DO YOU WISH TO REVIEW THE DESCRIPTION OF ',FNAME,'?',⍞,'
      ENTER Y OR N,'
[43] ⍝ IF RESPONSE IS NULL OR NOT Y, GO TO RESTART, ELSE GO ON
[44]  →RESTARTx\0=⍴B←⎕
[45]  →RESTARTx\'Y'≠B
[46] ⍝ DISPLAY HELP VARIABLE,THEN RESTART
[47]  RUN ROW
[48]  →RESTART
[49] ⍝ THREE RESTARTS; DISPLAY MESSAGE AND EXIT
[50] OUCH:⎕←'PLEASE SEE YOUR INSTRUCTOR FOR ASSISTANCE,'
[51]  →0
[52] ⍝ STOP AT STUDENT REQUEST; SET FLAG AND EXIT
[53] HALT:→0,Y←2
      ∇
      .
```

```
      ∇ TEST NNN;NAME;G;RANK;FNAME;T;RT;WT;QV;AV;TYPE;ARG;SS
[1]   ⍝ USED TO TEST DRILL ; CALLS RUNDRILL, TRY, SQUEEZE
[2]   ⍝ SET VARIABLES CORRESPONDING TO ROW
[3]    ROW←((⍎,MAT[;LESCOL])=NNN)/⍳(⍴MAT)[1]
[4]    NAME←,MAT[ROW;NAMECOL]
[5]    FNAME←SQUEEZE,MAT[ROW;FNAMECOL]
[6]    STM←MAT[ROW;STMCOL]
[7]    RANK←MAT[ROW;SVXACOL]
[8]    TYPE←MAT[ROW;MDCOL]
[9]    ARG←MAT[ROW;NKCOL]
[10]   T←⍳0
[11]  ⍝ SCALAR DRILL
[12]  SD:→V×⍳~STM∊SS←'+-×÷⌈⌊!|~⍳*<≤=≥>≠∨∧⍱⍲,⍴⍟'
[13]   →V×⍳((STM∊'+×⍴,')∧TYPE='M')∨(STM∊'⍳')∧TYPE='D'
[14]   1 RUNDRILL 1
[15]  ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[16]   →HALT×T=2
[17]  V:→VD×⍳(0=⎕NC NAME)
[18]  ⍝ VECTOR DRILL
[19]  VD:→X×⍳(~STM∊SS,'⍒⍋∊/\⌿⍀↑↓⌽⊖')∨((STM∊',+×')∧TYPE='M')∨(STM∊'/\⌿⍀')∧
       TYPE='D'
[20]   →VS×⍳(STM∊'⍳')
[21]   1 RUNDRILL 2
[22]  ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[23]   →HALT×T=2
[24]  VS:→VV×⍳(STM∊'⌽⍴⍒⍋')∧TYPE='M'
[25]   →X×⍳(STM∊'⍴↑↓⌽⊖')∨TYPE='M'
[26]   2 RUNDRILL 1
[27]  ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[28]   →HALT×T=2
[29]  VV:→X×⍳STM∊'⍳'
[30]   2 RUNDRILL 2
[31]  ⍝ IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
```

```
[32]  →HALTXY=2
[33] X;→MDX\(0=⎕NC NAME)
[34] MD;→AX\~(STM∈'F')∧TYPE='X'
[35]  1 RUNDRILL 3
[36] A IF FLAG INDICATES 'STOP' RESPONSE, GO TO HALT; ELSE GO ON
[37]  →HALTXY=2
[38] A;⎕←'TEST OK'
[39]  →0
[40] HALT;CR,'THIS TEST HAS BEEN HALTED,'
     ∇
     .
```

```
      ∇ Z←NEXTLESSON;I;V
[1]   ⍝FIND INDEX OF LAST LESSON IN VECTOR OF LESSON NUMBERS
[2]    I←(V←⍳,MAT[;LESCOL])⍳LASTLESSON
[3]   ⍝RETURN NEXT LESSON NUMBER IN SEQUENCE
[4]    Z←V[I+1]
      ∇
      .




      ∇ ORDERMAT;OM;ICOM
[1]   ⍝ SORTS THE VARIABLE 'MAT' IN ACSENDING ORDER OF LESSON NUMBER
[2]   ⍝ CREATE VECTOR OF LESSON NRS
[3]    OM←⍳,MAT[;⍳4]
[4]   ⍝ FIND INDEXES OF LESSONS IN ORDER
[5]    ICOM←⍋OM
[6]   ⍝ REORDER MAT ACCORDING TO INDEXES
[7]    MAT←MAT[ICOM;]
      ∇
      .




      ∇ Z←SQUEEZE STRING
[1]   ⍝ CALLED BY HELP, RUN, TEACH, SHOW, SCORE, DRILL
[2]   ⍝ DELETES BLANKS IN CHARACTER STRING ARGUMENT
[3]    Z←(' '≠,STRING)/,STRING
      ∇
      .
```

C. VARIABLES

1. Global variables

The following section contains all of the global variables in the TUTOR workspace except the text variables (part C) and the user's guide variables called by the START function (appendix A). MAT is the most important variable in TUTOR, as it is called on to cross-index symbols, lesson numbers, function names, and selected function characteristics. The variables with names ending in COL index the columns of MAT for use in the functions, so that these columns can be rearranged if necessary. CUES provides questions for use in the ASK function, and all of the two-letter, underlined variables (except CR) contain sets of multiple-choice answers for use with CUES. The other important global variable is LASTLESSON which is given a value by the SHOW function to indicate the last lesson completed. All other variables used in TUTOR functions are local variables at some level of the calling hierarchy.

	MAT							
101	-	M		N		S	NEGA	NEGATIVE_NUMBER
102	+	M		N		A	CONJ	CONJUGATE
103	+	D	N	N	A	A	PLUS	PLUS
104	-	M		N		A	CHAN	CHANGE_SIGN
105	-	D	N	N	A	A	MINU	MINUS
106	×	M		N		A	SIGN	SIGNUM
107	×	D	N	N	A	A	TIME	TIMES
108	÷	M		N		A	RECI	RECIPROCAL
109	÷	D	N	N	A	A	DIVI	DIVIDE
121	()	M		E		A	PARE	PARENS
122	''	M		K		V	QUOT	QUOTE
123	←	D	K	E	V	A	SPEC	SPECIFICATION
124	⍴	M		E		A	SHAP	SHAPE
125	⍴	D	N	E	V	A	RESH	RESHAPE
126	[]	D	E	N	A	A	BRAC	BRACKETS
127	,	M		E		A	RAVE	RAVEL
128	,	D	E	E	A	A	CATE	CATENATE
129	⍳	M		N		V	INDE	INDEX_GENERATOR
131	/	M		E		A	RED2	REDUCTION
132	⌿	M		E		A	RED1	REDUCTION
141	*	M		N		A	EXPO	EXPONENTIAL
142	*	D	N	N	A	A	POWE	POWER
143	⍟	M		N		A	NATU	NATURAL_LOG
144	⍟	D	N	N	A	A	LOGA	LOGARITHM
145	○	M		N		A	PITI	PI_TIMES
146	○	D	N	N	A	A	GEOM	GEOMETRIC
147	!	M		N		A	FACT	FACTORIAL
148	!	D	N	N	A	A	BINO	BINOMIAL
149	|	M		N		A	MAGN	MAGNITUDE
151	\	M		E		A	SCA2	SCAN
152	⍀	M		E		A	SCA1	SCAN
161	⌈	M		N		A	CEIL	CEILING
162	⌊	M		N		A	FLOO	FLOOR

163	⌈	D	N	N	A	A	MAXI	MAXIMUM
164	⌊	D	N	N	A	A	MINI	MINIMUM
165	|	D	N	N	A	A	RESI	RESIDUE
166	⍋	M		N		V	GRUP	GRADE_UP
167	⍒	M		N		V	GRAD	GRADE_DOWN
168	?	M		N		A	ROLL	ROLL
169	?	D	N	N	S	S	DEAL	DEAL
180	=	D	E	E	A	A	EQUA	EQUAL
181	≠	D	E	E	A	A	NOTE	NOT_EQUAL
182	<	D	N	N	A	A	LESS	LESS_THAN
183	≤	D	N	N	A	A	LTEQ	LESS_OR_EQUAL
184	≥	D	N	N	A	A	GTEQ	GREATER_OR_EQUAL
185	>	D	N	N	A	A	GREA	GREATER_THAN
191	~	M		B		A	NOT	NOT
192	∧	D	B	B	A	A	AND	AND
193	⍲	D	B	B	A	A	NAND	NAND
194	∨	D	B	B	A	A	OR	OR
195	⍱	D	B	B	A	A	NOR	NOR
196	∈	D	E	E	A	A	MEMB	MEMBER OF
201	⌽	M		E		A	REV2	REVERSE
202	⊖	M		E		S	REV1	REVERSE
203	⌽	D	N	E	A	A	ROT2	ROTATE
204	⊖	D	N	E	A	A	ROT1	ROTATE
205	⍉	M		E		A	TRAM	TRANSPOSE-MONADIC
206	⍉	D	N	E	V	A	TRAD	TRANSPOSE-DYADIC
211	↑	D	N	E	V	A	TAKE	TAKE
212	↓	D	N	E	V	A	DROP	DROP
213	/	D	B	E	V	A	COM2	COMPRESS
214	⌿	D	B	E	V	A	COM1	COMPRESS
215	\	D	B	E	V	A	EXP2	EXPAND
216	⍀	D	B	E	V	A	EXP1	EXPAND
219	,	D	E	E	A	A	LAMI	LAMINATE
231	⌹	M		N		X	MATI	MATRIX_INVERSE
232	⌹	D	N	N	X	X	MATD	MATRIX_DIVIDE

241	∘.	D	E	E	A	A	OUTE	OUTER_PRODUCT
242	.	D	E	E	A	A	INNE	INNER_PRODUCT
301	⍳	D	E	E	V	A	INDO	INDEX_OF
304	⍕	D	N	N	A	A	ENCO	ENCODE
305	⊥	D	N	N	A	A	DECO	DECODE
306	⍎	M		K		V	EXEC	EXECUTE
307	⍕	D	N	N	V	A	DYFO	FORMAT-DYADIC
308	⍕	M		E		A	FORM	FORMAT-MONADIC

```
        LESCOL
1 2 3 4
        .

        SYMCOL
8
        .

        MDCOL
12
        .

        NKCOL
16 18
        .

        SVXACOL
22 24
        .

        NAMECOL
28 29 30 31
        .

        FNAMECOL
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
        .
```

```
      CUES
'HOW MANY ARGUMENTS DOES THE ',FNAME,' FUNCTION TAKE?'

'DOES S1',SYM,'S2  EQUAL  S2',SYM,'S1?'

'IF ONE ARGUMENT IS A VECTOR, THE OTHER ARGUMENT MAY BE SCALAR OR VECTOR
     . (T/F)'
'IF ONE ARGUMENT IS A MATRIX, THE OTHER ARGUMENT MUST BE A SCALAR',CR,'
     OR A MATRIX OF THE SAME SHAPE. (T/F)'
'WHAT RANK CAN THE LEFT ARGUMENT OF ',FNAME,' BE?',RK

'WHAT RANK CAN THE RIGHT ARGUMENT OF ',FNAME,' BE?',RK

'DO THE ARGUMENTS ALWAYS HAVE TO HAVE THE SAME RANK?'

'CAN ',FNAME,' TAKE CHARACTER ARGUMENTS?'

'IF SO, WHICH ARGUMENT(S) CAN BE CHARACTER?',AR

'DOES ',FNAME,' TAKE BOOLEAN (1 OR 0) ARGUMENTS?'

'IF SO, WHICH ARGUMENT(S) MUST BE BOOLEAN?',AR

     ,AR

ENTER R FOR RIGHT,
     L FOR LEFT, OR
     B FOR BOTH
     ,RK

ENTER S FOR SCALAR
     V FOR VECTOR
     X FOR MATRIX
     A FOR ANY RANK
```

2. Text variables

The following global variables contain the basic text of the TUTOR course. They are printed in the same order as the lessons. The header above each variable is not part of the text; it lists the lesson number, the short title of the variables, and the full name of the function. The first column control characters are omitted here.

..... ¯ .....

THE SYMBOL ¯ (UPSHIFT 2) IS USED ONLY TO ENTER NEGATIVE NUMBERS,

IT CAN BE USED ONLY IN FRONT OF SCALARS (SINGLE NUMBERS),

NOT VARIABLE NAMES, ARITHMETIC EXPRESSIONS, OR ARRAYS,

TO CHANGE ARITHMETIC SIGN USE THE SYMBOL - (UPSHIFT +),

.....+.....

THE SYMBOL + IS USED FOR THE MONADIC CONJUGATE FUNCTION,

THIS IS AN IDENTITY FUNCTION WHICH RETURNS ITS ARGUMENT, AFTER

EVALUATION IF NECESSARY, IT TAKES NUMERIC ARGUMENTS OF ANY RANK,

EXAMPLE: +2 => 2 +¯3+2 => ¯1.

..... + ...

THE SYMBOL + IS USED FOR THE DYADIC PLUS FUNCTION.

THIS PERFORMS SIMPLE ADDITION ON NUMERIC ARGUMENTS OF ANY RANK.

FOR EXAMPLE: 2+2 => 4 SPACING IS NOT IMPORTANT.

TO ADD A SCALAR TO EACH ELEMENT OF A VECTOR, ENTER EITHER S+V OR V+S.

V1+V2 WILL ADD TWO VECTORS ELEMENT BY ELEMENT. FOR EXAMPLE:

```
   2+(3 4 5)  => 5 6 7        (3 4 5)+2  => 5 6 7
   (3 4 5)+(5 6 7)  => 8 10 12
```

IF THE TWO VECTORS BEING ADDED ARE NOT THE SAME LENGTH,

YOU WILL GET A 'LENGTH ERROR'.

MATRIX ADDITION IS DONE AS FOLLOWS:

S+M OR M+S WILL ADD THE SCALAR S TO EACH ELEMENT OF THE MATRIX M.

FOR EXAMPLE: IF M => 1 2 THEN 3+M => 4 5 AND M+3 => 4 5

3 4 6 7 6 7

M1+M2 WILL ADD THE MATRIX M1 TO THE MATRIX M2, ELEMENT BY ELEMENT.

FOR EXAMPLE: IF M1 => 1 2 AND M2 => 3 4 THEN M1+M2 => 4 6

3 4 5 6 8 10

M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT.

TRYING TO ADD A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... ¯ .....

THE SYMBOL - (UPSHIFT +) IS USED FOR THE MONADIC CHANGE SIGN FUNCTION.
THIS CHANGES THE ARITHMETIC SIGN OF ITS NUMERIC ARGUMENT OF ANY RANK.
FOR EXAMPLE: -(1 ¯2 3) => ¯1 2 ¯3.
NOTE THE DIFFERENCE BETWEEN - AND ¯ (UPSHIFT 2), WHICH INDICATES
NEGATIVE NUMBERS.

..... - .....

THE SYMBOL - (UPSHIFT +) IS USED FOR THE DYADIC MINUS FUNCTION,
IT PERFORMS SIMPLE SUBTRACTION ON NUMERIC ARGUMENTS OF ANY RANK,

FOR EXAMPLE: 5-2 ⇒ 3 ¯10-¯2 ⇒ ¯8

TO SUBTRACT A SCALAR FROM EACH ELEMENT OF A VECTOR, ENTER V-S,
TO SUBTRACT EACH ELEMENT OF A VECTOR FROM THE SAME SCALAR, ENTER S-V,
V1-V2 WILL SUBTRACT EACH ELEMENT OF V2 FROM THE CORRESPONDING
ELEMENT OF V1,
V1 AND V2 MUST BE THE SAME LENGTH OR YOU WILL GET A 'LENGTH ERROR',
FOR EXAMPLE: (3 4 5)-2 ⇒ 1 2 3 2-(3 4 5) ⇒ ¯1 ¯2 ¯3
(6 7 8)-(1 2 3) ⇒ 5 5 5

ENTER M-S TO SUBTRACT THE SCALAR S FROM EACH ELEMENT OF THE MATRIX M,
ENTER S-M TO DO THE OPPOSITE,
FOR EXAMPLE: IF M ⇒ 1 2 THEN 4-M ⇒ 3 2 AND M-2 ⇒ ¯1 0
3 4 1 0 1 2
M1-M2 SUBTRACTS MATRIX M2 FROM MATRIX M1, ELEMENT BY ELEMENT,
FOR EXAMPLE: IF M1 ⇒ 1 2 AND M2 ⇒ 3 4 THEN M2-M1 ⇒ 2 2
3 4 5 6 2 2
IF M1 AND M2 ARE NOT THE SAME SHAPE, A 'LENGTH ERROR' WILL RESULT,
TRYING TO SUBTRACT A VECTOR FROM A MATRIX OR VICE-VERSA WILL
PRODUCE A 'RANK ERROR',

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS,

..... x .....

THE SYMBOL x IS USED FOR THE MONADIC SIGNUM FUNCTION.

IT RETURNS THE ARITHMETIC SIGN OF THE ARGUMENT: 1 FOR POSITIVE NUMBERS,

¯1 FOR NEGATIVE NUMBERS, AND 0 FOR ZERO VALUES.

IT TAKES NUMERIC ARGUMENTS OF ANY RANK.

FOR EXAMPLE: x(3 ¯2) => 1 ¯1 x(3+2-5) => 0

..... X .....

THE SYMBOL x IS USED FOR THE DYADIC TIMES FUNCTION.

THIS PERFORMS SIMPLE MULTIPLICATION ON NUMERIC ARGUMENTS OF ANY RANK.

FOR EXAMPLE: 2x4 => 8 ORDER AND SPACING ARE NOT IMPORTANT.

TO MULTIPY EACH ELEMENT OF A VECTOR BY A SCALAR, ENTER SxV OR VxS.

V1xV2 RESULTS IN THE PRODUCT OF CORRESPONDING ELEMENTS OF V1 AND V2.

FOR EXAMPLE: 3x(1 2 3) => 3 6 9 (1 2 3)x3 => 3 6 9

(1 2 3)x(2 4 6) => 2 8 18

IF V1 AND V2 ARE NOT THE SAME LENGTH, YOU WILL GET A 'LENGTH ERROR'.

EITHER MxS OR SxM WILL MULTIPLY A MATRIX BY A SCALAR.

THE RESULT WILL BE A MATRIX THE SAME SHAPE AS M.

FOR EXAMPLE: IF M => 1 2 THEN 3xM => 3 6 AND Mx3 => 3 6

3 4 9 12 9 12

M1xM2 WILL RETURN THE PRODUCT OF CORRESPONDING ELEMENTS IN M1 AND M2.

M1 AND M2 MUST BE THE SAME SHAPE OR YOU WILL GET A 'LENGTH ERROR'.

FOR EXAMPLE: IF M1 => 1 2 AND M2 => 3 4 THEN M1xM2 => 3 8

3 4 5 6 15 24

IT IS IMPORTANT TO NOTE THAT MULTIPLYING MATRIXES WILL NOT RESULT IN

'MATRIX MULTIPLICATION' (INNER PRODUCT OR DOT PRODUCT).

FOR INFORMATION ON INNER PRODUCT, REQUEST HELP ON '.' (DOT OR PERIOD).

MULTIPLYING A VECTOR WITH A MATRIX WILL RESULT IN A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... ÷ .....

THE SYMBOL ÷ (UPSHIFT ÷) IS USED FOR THE MONADIC RECIPROCAL FUNCTION.

THIS RETURNS THE RESULT OF 1 DIVIDED BY THE ARGUMENT.

ITS ARGUMENT MUST BE NUMERIC AND CAN BE OF ANY RANK.

HOWEVER, IF THE ARGUMENT HAS VALUE 0, A 'DOMAIN ERROR' WILL RESULT.

FOR EXAMPLE: ÷(1 2 3) => 1 0.5 0.333333333

..... ÷ .....

THE SYMBOL ÷ (UPSHIFT X) IS USED FOR THE DYADIC DIVISION FUNCTION,
IT PERFORMS SIMPLE DIVISION ON NUMERIC ARGUMENTS OF ANY RANK,

FOR EXAMPLE: 6÷2 => 3 ¯10÷¯4 => 2.5
DIVIDING BY 0 RESULTS IN A 'DOMAIN ERROR', EXCEPT THAT 0÷0 => 1,

TO DIVIDE A SCALAR BY EACH ELEMENT OF A VECTOR, ENTER S÷V,
TO DIVIDE EACH ELEMENT OF A VECTOR BY THE SAME SCALAR, ENTER V÷S,
$V_1 \div V_2$ DIVIDES EACH ELEMENT OF V_1 BY THE CORRESPONDING ELEMENT OF V_2,
V_1 AND V_2 MUST BE THE SAME LENGTH OR YOU WILL GET A 'LENGTH ERROR',
FOR EXAMPLE: (3 4 5)÷2 => 1.5 2 2.5 3÷(3 4 5) => 1 .75 .6
(3 6 9)÷(1 2 3) => 3 3 3

ENTER S÷M TO DIVIDE THE SCALAR S BY EACH ELEMENT OF THE MATRIX M,
M÷S WILL DIVIDE EACH ELEMENT OF M BY S, BOTH OPERATIONS RESULT IN
A MATRIX THE SAME SHAPE AS M,

FOR EXAMPLE: IF M => 1 2 THEN 2÷M => 2 1 AND M÷2 => 0.5 1
4 5 0.5 0.4 2 2.5

$M_1 \div M_2$ DIVIDES MATRIX M_1 BY MATRIX M_2, ELEMENT BY ELEMENT,

FOR EXAMPLE: IF M_1 => 2 4 AND M_2 => 1 2 THEN $M_1 \div M_2$ => 2 2
6 8 3 4 2 2

IF M_1 AND M_2 ARE NOT THE SAME SHAPE, A 'LENGTH ERROR' WILL RESULT,
TRYING TO DIVIDE A MATRIX BY A VECTOR OR VICE-VERSA WILL PRODUCE A
'RANK ERROR',

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS,

..... () .....

PARENTHESES ARE USED TO GROUP SYMBOLS FOR CORRECT PROCESSING BY
THE APL INTERPRETER.

APL DIFFERS FROM EVERY OTHER COMPUTER LANGUAGE IN THE WAY IT
DETERMINES WHICH OPERATION IS PERFORMED FIRST.

IT SIMPLY READS FROM RIGHT TO LEFT--'BACKWARDS'--AND PERFORMS EACH
OPERATION AS IT GOES ALONG.

FOR EXAMPLE: 3×2+1 ⇒ 9 BECAUSE 2+1 ⇒ 3 AND 3×3 ⇒ 9
1+2×3 ⇒ 7 BECAUSE 2×3 ⇒ 6 AND 1+6 ⇒ 7

PARENTHESES ARE USED TO OVERRIDE THIS RIGHT-TO-LEFT RULE.

FOR EXAMPLE: (3×2)+1 ⇒ 7 AND (1+2)×3 ⇒ 9

EXPRESSIONS WITHIN PARENTHESES ARE EVALUATED FIRST--
ALSO FROM RIGHT TO LEFT.

USING THE RIGHT-TO-LEFT RULE CAN ELIMINATE PARENTHESES,
BUT USING PARENTHESES GENEROUSLY CAN ELIMINATE CONFUSION.

..... ' .....

THE QUOTE MARK ' (UPSHIFT K) IS USED TO ENCLOSE CHARACTER DATA.

APL ASSUMES THAT LETTERS (OR COMBINATIONS OF LETTERS AND NUMBERS)

ARE VARIABLE OR FUNCTION NAMES UNLESS THEY ARE ENCLOSED IN QUOTES.

FOR EXAMPLE: IN THIS TUTORIAL,

```
   HOW    =>    (THE VARIABLE NAMED HOW)
  'HOW'   =>     HOW
```

YOU CAN ALSO USE SYMBOLS AND NUMBERS AS CHARACTER DATA.

..... ← .....

THE SYMBOL ← IS USED FOR SPECIFICATION,

IT ASSIGNS THE VALUE OF THE RIGHT ARGUMENT TO THE VARIABLE NAME WHICH IS THE LEFT ARGUMENT,

THE VARIABLE NAME CAN BE ANY COMBINATION OF LETTERS AND NUMBERS (NO SYMBOLS AND NO SPACES) WHICH BEGINS WITH A LETTER, LENGTH IS EFFECTIVELY UNLIMITED,

THE RIGHT ARGUMENT CAN BE EITHER CHARACTER OR NUMERIC, OF ANY RANK, TO SPECIFY A CHARACTER VARIABLE, ENCLOSE THE RIGHT ARGUMENT IN SINGLE QUOTES (') UNLESS IT IS ANOTHER CHARACTER VARIABLE,

NOTE THAT THE VARIABLE CAN BE USED IN THE SAME LINE AS THE VALUE IS SPECIFIED, FOR EXAMPLE:

```
      X←3+Y←2   RESULTS IN X HAVING VALUE 2 AND Y HAVING VALUE 5
      X←Y←Z←1   RESULTS IN X, Y, AND Z ALL HAVING VALUE 1

      V←1 2 3   RESULTS IN V HAVING VECTOR VALUE (1 2 3)
      V1←'APL'  RESULTS IN V1 BEING A CHARACTER VECTOR: APL
      V2←V1     RESULTS IN V2 BEING A CHARACTER VECTOR: APL
```

SEE HELP ON ⍴ (UPSHIFT R) FOR INFORMATION ON SPECIFYING MATRIX VARIABLES,

..... ⍴ .....

THE SYMBOL ⍴ IS USED FOR THE MONADIC SHAPE FUNCTION.

THIS RETURNS A DESCRIPTION OF THE SIZE OF ITS ARGUMENT, WHICH CAN BE NUMERIC OR CHARACTER, OF ANY RANK.

THE SYMBOL CAN BE USED TWICE (⍴⍴A) TO RETURN THE RANK OF ITS ARGUMENT.

IF THE ARGUMENT IS A SCALAR, ⍴S RETURNS NO NUMBER (AN EMPTY VECTOR), BECAUSE A SCALAR HAS NO DIMENSION. ⍴⍴S ⇒ 0

ENTERING ⍴V RETURNS ONE NUMBER WHICH REPRESENTS THE LENGTH OF THE VECTOR'S ONE DIMENSION. ⍴⍴V ⇒ 1

THE SHAPE OF MATRIXES IS EXPRESSED AS A TWO-ELEMENT VECTOR, THE FIRST ELEMENT EQUALS THE NUMBER OF ROWS (THE LENGTH OF THE FIRST DIMENSION), AND THE SECOND ELEMENT EQUALS THE NUMBER OF COLUMNS (THE LENGTH OF THE SECOND DIMENSION).

FOR EXAMPLE: IF M ⇒ 1 2 3 THEN ⍴M ⇒ 2 3 AND ⍴⍴M ⇒ 2
4 5 6

FOR HIGHER-LEVEL ARRAYS, ⍴A WILL BE A VECTOR WITH AS MANY ELEMENTS AS A HAS DIMENSIONS. THE NUMBER OF COLUMNS WILL ALWAYS BE THE LAST ELEMENT OF ⍴A. THE NUMBER OF ROWS WILL ALWAYS BE THE NEXT-TO-LAST ELEMENT, PRECEDED BY THE NUMBER OF 'PAGES' OR PLANES, PRECEDED BY THE NUMBER OF 'BOOKS' OR SPACES, AND SO ON.

FOR EXAMPLE: IF D IS A 4-DIMENSIONAL ARRAY, ⍴⍴D ⇒ 4.
IF ⍴D ⇒ 5 3 4 2 THEN D HAS 5 BOOKS, EACH WITH 3 PAGES, EACH OF WHICH HAS 4 ROWS AND 2 COLUMNS. (THIS WOULD BE DISPLAYED AS FIFTEEN SUCCESSIVE 4X2 MATRIXES--THE FIRST THREE BEING THE FIRST BOOK, THE SECOND THREE BEING THE SECOND BOOK, ETC.)

..... ⍴ .....

THE SYMBOL ⍴ IS USED FOR THE DYADIC RESHAPE FUNCTION,
THIS TAKES THE ELEMENTS OF THE RIGHT ARGUMENT AND REARRANGES THEM ACCORDING TO THE SHAPE SPECIFIED BY THE LEFT ARGUMENT,

THE LEFT ARGUMENT MUST BE AN INTEGER SCALAR OR A VECTOR OF INTEGER ELEMENTS,

THE RIGHT ARGUMENT CAN BE NUMERIC OR CHARACTER, OF ANY RANK,

IF THE LEFT ARGUMENT IS A SCALAR, THE RESULT WILL BE A VECTOR WITH LENGTH EQUAL TO THE SCALAR, CONSISTING OF THE ELEMENTS OF THE RIGHT ARGUMENT, TAKEN IN ORDER TOP LEFT TO BOTTOM RIGHT,

WHEN THERE ARE NOT ENOUGH ELEMENTS IN THE RIGHT ARGUMENT, THE FUNCTION WILL START OVER AND TAKE THE ELEMENTS IN ORDER AGAIN, AS MANY TIMES AS NECESSARY TO FILL THE VECTOR,

FOR EXAMPLE; 4⍴1 ⇒ 1 1 1 1

4⍴('AB') ⇒ ABAB

IF THERE ARE TOO MANY ELEMENTS ON THE RIGHT, THE EXTRAS ARE OMITTED,

FOR EXAMPLE; 2⍴(1 2 3 4) ⇒ 1 2

IF THE LEFT ARGUMENT IS A VECTOR, THE ELEMENTS INDICATE, IN ORDER, LENGTH OF EACH DIMENSION IN THE RESULT,

FOR EXAMPLE; 2 2⍴1 2 3 4 ⇒ 1 2
3 4

THIS IS ONE WAY TO ENTER MATRIXES OR ARRAYS OF HIGHER DIMENSION,

USING A MATRIX OR HIGHER-LEVEL ARRAY AS THE LEFT ARGUMENT OF RESHAPE WILL RESULT IN A 'RANK ERROR',

..... [] .....

DYADIC SQUARE BRACKETS ARE USED TO INDEX THE ELEMENTS OF AN ARRAY,
THEY CAN BE THOUGHT OF AS ENCLOSING SUBSCRIPTS.

THE LEFT ARGUMENT (LEFT OF THE BRACKETS) CAN BE A NUMERIC OR CHARACTER ARRAY OF ANY RANK, BUT NOT A SCALAR. INDEXING A SCALAR WILL RESULT IN A 'RANK ERROR'.

THE RIGHT ARGUMENT (BETWEEN THE BRACKETS) MUST BE ONE OR MORE INTEGER SCALARS OR VECTORS, SEPARATED BY SEMICOLONS (;). THERE MUST BE AS MANY SCALARS/VECTORS AS THERE ARE DIMENSIONS IN THE LEFT ARGUMENT, OR A 'RANK ERROR' WILL RESULT.

ANY NUMERIC EXPRESSIONS WITHIN THE BRACKETS WILL BE EVALUATED AS IF THE BRACKETS AND SEMICOLONS ARE PARENTHESES.

BRACKETS CAN ALSO BE USED FOLLOWING THESE SYMBOLS: ,/\⌿⍀⌽⊖
FOR MORE INFORMATION, SEE HELP ON THE APPROPRIATE SYMBOL.

FOR EXAMPLE: IF V IS A VECTOR WITH VALUE (3 4 5 6),

```
V[1]  ⇒  3     V[1 2]  ⇒  3 4    V[1+2]  ⇒  5    V[1]+V[2]  ⇒ 7
'A PROGRAMMING LANGUAGE'[1 3 15]   ⇒   APL
```

THERE IS ONLY ONE SCALAR OR VECTOR WITHIN THE BRACKETS BECAUSE VECTORS HAVE ONLY ONE DIMENSION.

MATRIXES REQUIRE TWO SUBSCRIPTS--TWO SCALARS OR VECTORS, SEPARATED BY A SEMICOLON(;), WITHIN THE BRACKETS.

THE FIRST SUBSCRIPT REFERS TO THE ROW, THE SECOND TO THE COLUMN.

FOR EXAMPLE: IF M ⇒ 1 2 3 THEN M[1;1] ⇒ 1 AND M[1 3;2] ⇒ 2 8
4 5 6
7 8 9

TO SELECT AN ENTIRE ROW (OR ROWS), ENTER THE ROW SUBSCRIPT FOLLOWED BY A SEMICOLON, WITH NO COLUMN SUBSCRIPT.

FOR EXAMPLE: M[1;] ⇒ 1 2 3

FOR ENTIRE COLUMN(S), LEAVE THE ROW SUBCRIPT BLANK AND ENTER A SEMICOLON FOLLOWED BY THE COLUMN SUBSCRIPT(S).

```
FOR EXAMPLE;   IF M1  =>  FOR     THEN  M1[;1]   =>   FYI
                         YOUR
                         INFO
```

NOTICE THAT ONE ROW OR ONE COLUMN OF A MATRIX IS SIMPLY A VECTOR, NOT A 1xC OR Rx1 MATRIX. A SINGLE ELEMENT OF A MATRIX IS A SCALAR, NOT A 1x1 MATRIX OR A VECTOR OF LENGTH 1.

THE USE OF BRACKETS CAN BE EXTENDED TO HIGHER DIMENSIONS BY ADDING A SEMICOLON AND A SCALAR OR VECTOR FOR EACH DIMENSION.

FOR EXAMPLE, IF ⍴M => 2 3 4 5, THE SECOND BOOK, THIRD PAGE, FIRST ROW (ALL COLUMNS) CAN BE SELECTED BY ENTERING M[2;3;1;].

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, , ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

THE SYMBOL , IS USED FOR THE MONADIC RAVEL FUNCTION.

THIS FUNCTION CHANGES ITS ARGUMENT TO THE FORM OF A VECTOR. IT TAKES EITHER NUMERIC OR CHARACTER ARGUMENTS OF ANY RANK.

TO CHANGE A SCALAR TO A VECTOR OF LENGTH 1, ENTER ,S.

FOR EXAMPLE: ⍴⍴3 ⍝ 0 BUT ⍴⍴,3 ⍝ 1

THIS CAN BE USEFUL TO ENSURE THAT A VARIABLE IS A VECTOR WHEN USING A SCALAR WOULD LEAD TO AN ERROR (E.G., INDEXING).

VECTORS REMAIN UNCHANGED BY RAVELLING.

MATRIXES ARE CHANGED INTO A SINGLE LONG VECTOR, STARTING WITH THE TOP ROW AND READING ACROSS EACH ROW FROM LEFT TO RIGHT.

FOR EXAMPLE: IF M1 ⍝ 1 2 THEN ,M1 ⍝ 1 2 3 4
3 4

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... , ...

THE SYMBOL , IS USED FOR THE DYADIC CATENATE FUNCTION,
THIS FUNCTION WILL COMBINE TWO SCALARS OR ARRAYS INTO A SINGLE ARRAY,
IT TAKES EITHER NUMERIC OR CHARACTER ARGUMENTS,

ENTERING S1,S2 WILL RESULT IN A VECTOR (S1 S2), FOR EXAMPLE;
V←1,2 WILL ASSIGN THE VECTOR VALUE (1 2) TO THE VARIABLE V,

VECTORS CAN EASILY BE EXTENDED BY CATENATING S,V OR V,S OR V,V,
FOR EXAMPLE; IF V ⇒ 1 2 THEN
V,3 ⇒ 1 2 3 0,V ⇒ 0 1 2 V,V ⇒ 1 2 1 2

CATENATING A SCALAR TO A MATRIX WILL RESULT IN A MATRIX WITH
ONE MORE COLUMN THAN THE MATRIX ARGUMENT, ALL ENTRIES IN THIS
COLUMN WILL HAVE THE VALUE OF THE SCALAR, FOR EXAMPLE;

IF M ⇒ 1 2 THEN 5,M ⇒ 5 1 2 AND M,5 ⇒ 1 2 5
3 4 5 3 4 3 4 5

CATENATING TWO MATRIXES WILL RESULT IN A MATRIX CONTAINING THE TWO
ORIGINAL MATRIXES 'SIDE BY SIDE',
BOTH MATRIXES MUST HAVE THE SAME NUMBER OF ROWS OR A 'LENGTH ERROR'
WILL RESULT,
FOR EXAMPLE; IF M1 ⇒ 1 2 AND M2 ⇒ 5 6
3 4 7 8
THEN M1,M2 ⇒ 1 2 5 6 AND M2,M1 ⇒ 5 6 1 2
3 4 7 8 7 8 3 4
TRYING TO CATENATE A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR',

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS,
FOR MORE INFORMATION ON DIFFERENT WAYS TO JOIN ARRAYS, SEE
MENU FOR LAMINATE,

..... \ .....

THE SYMBOL \ (UPSHIFT I) IS USED FOR THE MONADIC INDEX GENERATING FUNCTION.

IT IS USED WITH A SINGLE NON-NEGATIVE INTEGER ARGUMENT (SCALAR, OR VECTOR OF LENGTH 1).

IT RETURNS A VECTOR OF INTEGERS, IN ORDER, BEGINNING WITH THE INDEX ORIGIN, AND ENDING WITH THE ARGUMENT.

FOR EXAMPLE: \3 => 1 2 3 \0 => (AN EMPTY VECTOR)

THE INDEX ORIGIN IS NORMALLY (BY DEFAULT) 1.

USING \ WITH A NON-INTEGER, NEGATIVE OR ARRAY ARGUMENT WILL PRODUCE A 'DOMAIN ERROR'.

..... / .....

THE SYMBOL / IS USED FOLLOWING ANOTHER FUNCTION SYMBOL AND
PRECEDING A SINGLE ARGUMENT TO PRODUCE THE MIXED FUNCTION
CALLED REDUCTION.

REDUCTION MAY BE USED WITH NUMERIC ARGUMENTS OF ANY RANK.

REDUCTION MAY ALSO BE USED WITH CHARACTER ARGUMENTS IF THE FUNCTION
IT IS BEING COMBINED WITH ACCEPTS CHARACTER ARGUMENTS.

USING / WITH A SCALAR SIMPLY RETURNS THE SCALAR.

WHEN USED WITH A VECTOR, REDUCTION GIVES THE SAME EFFECT AS INSERTING
THE OTHER FUNCTION SYMBOL BETWEEN EACH VALUE OF THE VECTOR.

FOR EXAMPLE:

+/(1 2 3) ⍝ 6 BECAUSE 1+2+3 ⍝ 6

×/(2 3 4) ⍝ 24 BECAUSE 2×3×4 ⍝ 24

WHEN USING REDUCTION, ALWAYS REMEMBER APL READS RIGHT TO LEFT:

-/(1 2 3) ⍝ 2 BECAUSE 1-(2-3) ⍝ 2

(NOTICE THIS PRODUCES THE EFFECT OF SUMMATION WITH ALTERNATING SIGNS.)

WHEN USED WITH A MATRIX, / WILL PERFORM AS IF EACH COLUMN IS
A UNIT. IT WILL ADD THE CORRESPONDING ELEMENTS OF EACH COLUMN,
SO THAT THE ANSWER WILL BE A VECTOR WITH AS MANY ELEMENTS AS
THERE WERE ROWS IN THE ARGUMENT.

FOR EXAMPLE: IF ⍴M ⍝ 2 4 THEN ⍴+/M ⍝ 2

APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE COLUMNS.
SINCE COLUMNS ARE THE SECOND OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL / WORKS 'OVER' THE LAST DIMENSION.

TO REDUCE OVER THE FIRST DIMENSION (E.G. ROWS OF A MATRIX), USE THE
SYMBOL ⌿ (ALT /), IN PLACE OF /.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS.
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND

COLUMNS, / WILL REDUCE OVER THE COLUMNS, ⌿ OVER THE PAGES,

TO REDUCE OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN BRACKETS FOLLOWING THE ⌿ (OR THE /) SYMBOL.

FOR EXAMPLE, TO PLUS REDUCE OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY CALLED AA, ENTER +⌿[2]AA OR +/[2]AA. TO PLUS REDUCE OVER THE PAGES, ENTER +⌿AA OR +/[1]AA. TO PLUS REDUCE OVER THE COLUMNS, ENTER +/AA OR +⌿[3]AA.

THE SHAPE OF THE ANSWER WILL ALWAYS BE THE SAME AS THE SHAPE OF THE ARGUMENT OMITTING THE DIMENSION REDUCED OVER.

FOR EXAMPLE; IF ⍴A ⇒ 3 4 5

THEN ⍴+/A ⇒ 3 4 ⍴+/[2]A ⇒ 3 5 ⍴+⌿A ⇒ 4 5

..... ⌿ .....

THE SYMBOL ⌿ (ALT /) IS USED FOLLOWING ANOTHER FUNCTION SYMBOL AND
PRECEDING A SINGLE ARGUMENT TO PRODUCE THE MIXED FUNCTION
CALLED REDUCTION.

REDUCTION MAY BE USED WITH NUMERIC ARGUMENTS OF ANY RANK.

REDUCTION MAY ALSO BE USED WITH CHARACTER ARGUMENTS IF THE FUNCTION
IT IS BEING COMBINED WITH ACCEPTS CHARACTER ARGUMENTS.

USING ⌿ WITH A SCALAR SIMPLY RETURNS THE SCALAR.

WHEN USED WITH A VECTOR, REDUCTION GIVES THE SAME EFFECT AS INSERTING
THE OTHER FUNCTION SYMBOL BETWEEN EACH VALUE OF THE VECTOR.

FOR EXAMPLE;

+⌿(1 2 3) ⍝ 6 BECAUSE 1+2+3 ⍝ 6

×⌿(2 3 4) ⍝ 24 BECAUSE 2×3×4 ⍝ 24

WHEN USING REDUCTION, ALWAYS REMEMBER APL READS RIGHT TO LEFT;

-⌿(1 2 3) ⍝ 2 BECAUSE 1-(2-3) ⍝ 2

(NOTICE THIS PRODUCES THE EFFECT OF SUMMATION WITH ALTERNATING SIGNS.)

WHEN USED WITH A MATRIX, ⌿ WILL PERFORM AS IF EACH ROW IS
A UNIT. IT WILL ADD THE CORRESPONDING ELEMENTS OF EACH ROW,
SO THAT THE ANSWER WILL BE A VECTOR WITH AS MANY ELEMENTS AS
THERE WERE COLUMNS IN THE ARGUMENT.

FOR EXAMPLE; IF ⍴M ⍝ 2 4 THEN ⍴+⌿M ⍝ 4

APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE ROWS.
SINCE ROWS ARE THE FIRST OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL ⌿ WORKS 'OVER' THE FIRST DIMENSION.

TO REDUCE OVER THE LAST DIMENSION (E.G. COLUMNS OF A MATRIX), USE THE
SYMBOL / IN PLACE OF ⌿.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS.
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND

COLUMNS, / WILL REDUCE OVER THE COLUMNS, ⌿ OVER THE PAGES,
TO REDUCE OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN
BRACKETS FOLLOWING THE ⌿ (OR THE /) SYMBOL.
FOR EXAMPLE, TO PLUS REDUCE OVER THE ROWS OF A THREE-DIMENSIONAL
ARRAY CALLED AA, ENTER +⌿[2]AA OR +/[2]AA. TO PLUS REDUCE
OVER THE PAGES, ENTER +⌿AA OR +/[1]AA. TO PLUS REDUCE OVER
THE COLUMNS, ENTER +/AA OR +⌿[3]AA.
THE SHAPE OF THE ANSWER WILL ALWAYS BE THE SAME AS THE SHAPE OF THE
ARGUMENT OMITTING THE DIMENSION REDUCED OVER.
FOR EXAMPLE; IF ⍴A ⇒ 3 4 5
THEN ⍴+⌿A ⇒ 4 5 ⍴+/[2]A ⇒ 3 5 ⍴+/A ⇒ 3 4

..... * .....

THE SYMBOL * (UPSHIFT P) IS USED FOR THE MONADIC EXPONENTIAL FUNCTION,
IT RETURNS THE VALUE OF THE CONSTANT E TO THE POWER OF THE ARGUMENT,
IT TAKES NUMERIC ARGUMENTS OF ANY RANK,

```
FOR EXAMPLE:   *0  => 1          *1  => 2.718281828
               *.5 => 1.648721271
               *(3 ¯2) => 20.08553692  .1353352832
```

..... * .....

THE SYMBOL * (UPSHIFT P) IS USED FOR THE DYADIC POWER FUNCTION,
THIS RETURNS THE VALUE(S) OF THE LEFT ARGUMENT RAISED TO THE POWER OF
RIGHT ARGUMENT, ARGUMENTS ARE NUMERIC OF ANY RANK,
A NON-INTEGER RIGHT ARGUMENT CAN BE USED TO OBTAIN ROOTS,
FOR EXAMPLE: 2*3 => 8 3*0 => 1 4*.5 => 2

TO RAISE EACH ELEMENT OF A VECTOR TO THE SAME POWER, ENTER V*P,
TO RAISE A SCALAR TO A SERIES OF POWERS REPRESENTED BY A VECTOR,
ENTER S*V,
V1*V2 RESULTS IN THE ELEMENTS OF V1 BEING RAISED TO THE POWER OF
THE CORRESPONDING ELEMENT IN V2,
V1 AND V2 MUST HAVE THE SAME NUMBER OF ELEMENTS OR A 'LENGTH ERROR'
WILL RESULT,
FOR EXAMPLE: 3*(1 2 3) => 3 9 27 (1 2 3)*3 => 1 8 27
(2 3 4)*(5 4 3) => 32 81 64

TO RAISE EACH ELEMENT OF THE MATRIX M TO THE SAME POWER S,
ENTER M*S, TO RAISE THE SCALAR S TO THE POWER OF EACH ELEMENT
IN THE MATRIX M, ENTER S*M,
THE RESULT WILL BE A MATRIX THE SAME SHAPE AS M,
FOR EXAMPLE: IF M => 1 2 THEN 3*M => 3 9 AND M*3 => 1 8
3 4 27 81 27 64
M1*M2 RESULTS IN THE ELEMENTS OF M1 RAISED TO THE POWER
OF THE CORRESPONDING ELEMENT IN M2,
FOR EXAMPLE: IF M1 => 1 2 AND M2 => 4 3 THEN M1*M2 => 1 8
3 4 2 1 9 4
IF M1 AND M2 ARE NOT THE SAME SHAPE, A 'LENGTH ERROR' WILL OCCUR,
IF ONE ARGUMENT IS A VECTOR WHILE THE OTHER ARGUMENT IS A MATRIX,
THE RESULT WILL BE A 'RANK ERROR',

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS,

..... ⍟ .....

THE SYMBOL ⍟ (ALT 8) IS USED FOR THE MONADIC NATURAL LOG FUNCTION.

IT RETURNS THE VALUE OF THE POWER TO WHICH THE CONSTANT E MUST
BE RAISED TO EQUAL THE ARGUMENT. IN OTHER WORDS, ENTERING
⍟S WILL SOLVE THE APL EQUATION *N = S FOR N.

IT TAKES ONLY POSITIVE NUMERIC ARGUMENTS OF ANY RANK. USING ⍟ WITH
ZERO OR WITH NEGATIVE ARGUMENTS WILL RESULT IN A 'DOMAIN ERROR'.

```
FOR EXAMPLE:   ⍟1   ⇒  0          ⍟2.718281828  ⇒  1
               ⍟1.648721271  ⇒  0.5
FOR EXAMPLE:   ⍟(3 2)  ⇒  1.098612289  0.6931471806
```

..... ⍟ .....

THE SYMBOL ⍟ (ALT 8) IS USED FOR THE DYADIC LOGARITHM FUNCTION.
THIS RETURNS THE LOGARITHM OF THE RIGHT ARGUMENT IN THE BASE OF THE LEFT ARGUMENT. THAT IS, IT RETURNS THE VALUE OF THE POWER TO WHICH THE LEFT ARGUMENT MUST BE RAISED TO EQUAL THE RIGHT ARGUMENT.
ENTER L⍟R TO SOLVE THE EQUATION L*N = R FOR N.
ARGUMENTS ARE POSITIVE NUMERIC OF ANY RANK. USING ⍟ WITH ZERO OR WITH NEGATIVE NUMBERS WILL RESULT IN A 'DOMAIN ERROR'. ENTERING 1⍟S WHERE S IS ANY NUMBER EXCEPT 1 WILL ALSO PRODUCE A 'DOMAIN ERROR'.

FOR EXAMPLE: 10⍟100 ⇒ 2 2⍟1024 ⇒ 10 9⍟1 ⇒ 0

TO OBTAIN THE LOG OF EACH ELEMENT OF A VECTOR IN THE SAME BASE, ENTER B⍟V.
TO DETERMINE THE LOG OF A SCALAR IN A VECTOR OF DIFFERENT BASES, ENTER V⍟S.
V1⍟V2 RESULTS IN THE LOG OF EACH ELEMENT IN V2 IN THE BASE OF THE CORRESPONDING ELEMENT IN V1.
V1 AND V2 MUST HAVE THE SAME NUMBER OF ELEMENTS OR A 'LENGTH ERROR' WILL RESULT.
FOR EXAMPLE: 3⍟(3 9 27) ⇒ 1 2 3 (2 4)⍟16 ⇒ 4 2
(2 3 5)⍟(4 9 25) ⇒ 2 2 2

ANALOGOUS RESULTS ARE OBTAINED FOR MATRIXES AND HIGHER LEVEL ARRAYS.
UNLESSS ONE ARGUMENT IS A SCALAR, BOTH ARGUMENTS MUST BE THE SAME SHAPE. OTHERWISE, A 'RANK ERROR' OR 'LENGTH ERROR' WILL RESULT.

..... ⍜ .....

THE SYMBOL ⍜ (UPSHIFT O) IS USED FOR THE MONADIC PI TIMES FUNCTION,
IT RETURNS THE VALUE OF THE CONSTANT PI TIMES THE ARGUMENT,
IT TAKES NUMERIC ARGUMENTS OF ANY RANK,

```
FOR EXAMPLE:   ⍜0   =>  0          ⍜1   =>  3.141592654
               ⍜.5  =>  1.570796327
               ⍜(3 ¯2)  =>  9.424777961 ¯6.283185307
```

..... ⍜ .....

THE SYMBOL ⍜ (UPSHIFT O) IS USED FOR THE DYADIC GEOMETRIC FUNCTIONS. THESE INCLUDE THE MAIN TRIGINOMETRIC FUNCTIONS. THE LEFT ARGUMENT, WHICH MUST BE AN INTEGER FROM ¯7 TO 7, DETERMINES WHICH FUNCTION IS CALLED. THE RIGHT ARGUMENT, WHICH CAN BE A NUMERIC ARRAY OF ANY RANK, REPRESENTS THE VALUE OF AN ANGLE IN RADIANS.

THE FOLLOWING FUNCTIONS ARE REPRESENTED:

```
   0⍜Y  =>  COS (ARCSIN Y)  OR  SIN (ARCCOS Y) WHERE |Y|<1
   1⍜Y  =>  SIN Y                ¯1⍜Y  =>  ARCSIN Y WHERE |Y|<1
   2⍜Y  =>  COS Y                ¯2⍜Y  =>  ARCCOS Y WHERE |Y|<1
   3⍜Y  =>  TAN Y                ¯3⍜Y  =>  ARCTAN Y
   4⍜Y  =>  COSH (ARCSINH Y)     ¯4⍜Y  =>  SINH (ARCCOSH Y) WHERE |Y|>1
   5⍜Y  =>  SINH Y               ¯5⍜Y  =>  ARCSINH Y
   6⍜Y  =>  COSH Y               ¯6⍜Y  =>  ARCCOSH Y WHERE Y>1
   7⍜Y  =>  TANH Y               ¯7⍜Y  =>  ARCTANH Y WHERE |Y|<1
```

OTHER FUNCTIONS, SUCH AS SECANT, MUST BE COMPUTED BY FORMULA.

FOR EXAMPLE: (REMEMBER ⍜1 EQUALS PI TIMES 1)

```
   100  =>  0          1001  =>  0          100.5  =>  1
   200  =>  1          2001  =>  ¯1         200.5  =>  1.743934249E¯16
```

NOTICE THAT SOMETIMES A VERY SMALL NUMBER WILL APPEAR INSTEAD OF ZERO, OR A VERY LARGE NUMBER WILL APPEAR INSTEAD OF A DOMAIN ERROR. TECHNICALLY, THE TANGENT OF ONE-HALF PI DOES NOT EXIST, BUT:

```
   300.5  =>  5.734161139E15     WATCH OUT FOR THIS!
```

..... ! .....

THE SYMBOL ! (ALT +) IS USED FOR THE MONADIC FACTORIAL FUNCTION,
IT RETURNS THE FACTORIAL OF NON-NEGATIVE NUMERIC ARGUMENTS OF ANY RANK,
FOR EXAMPLE: !1 3 0 => 1 6 1

USING NON-INTEGER ARGUMENTS PRODUCES THE GAMMA FUNCTION OF THE
ARGUMENT+1.
FOR EXAMPLE: !.5 => .8862269255 (GAMMA OF 1.5)

..... ! .....

THE SYMBOL ! (ALT +) IS USED FOR THE DYADIC BINOMIAL FUNCTION.

IT TAKES NUMERIC ARGUMENTS OF ANY RANK.

FOR POSITIVE INTEGER ARGUMENTS, L!R WILL RETURN THE NUMBER OF WAYS OF TAKING R OBJECTS L AT A TIME.

THIS IS THE BINOMIAL COEFFICIENT OF R OVER L, COMPUTED (!R)÷!Lx!(R-L).

WITH NEGATIVE OR NON-INTEGER ARGUMENTS, THE BINOMIAL FUNCTION RETURNS A VALUE BASED ON THE BETA FUNCTION.

FOR EXAMPLE: 3!4 => 4 BUT 4!3 => 0

BECAUSE THERE ARE FOUR WAYS OF TAKING FOUR OBJECTS THREE AT A TIME, BUT THERE IS NO WAY TO TAKE THREE OBJECTS FOUR AT A TIME.

THESE ANSWERS ARE BASED ON CONVENTION:

0!0 => 1 0!42 => 1 42!42 => 1

WHEN A SCALAR ARGUMENT IS PAIRED WITH A VECTOR ARGUMENT, THE FUNCTION COMBINES THE SCALAR WITH EACH ELEMENT OF THE VECTOR IN TURN.

2!(3 4 5) => 3 6 10 (3 4 5)!5 => 10 5 1

CORRESPONDING ELEMENTS OF TWO VECTOR ARGUMENTS ARE COMBINED.

(3 4 5)!(5 6 7) => 10 15 21

V1 AND V2 MUST HAVE THE SAME NUMBER OF ELEMENTS OR A 'LENGTH ERROR' WILL RESULT.

WHEN A SCALAR ARGUMENT IS PAIRED WITH A MATRIX ARGUMENT, THE FUNCTION COMBINES THE SCALAR WITH EACH ELEMENT OF THE MATRIX IN TURN.

FOR EXAMPLE: IF M => 1 2 THEN 3!M => 0 0 AND M!3 => 3 3
3 4 1 4 1 0

CORRESPONDING ELEMENTS OF TWO MATRIX ARGUMENTS ARE COMBINED.

FOR EXAMPLE: IF M1 => 1 2 AND M2 => 3 4 THEN M1!M2 => 3 6
3 4 5 6 10 15

IF M1 AND M2 ARE NOT THE SAME SHAPE, A 'LENGTH ERROR' WILL OCCUR.

IF ONE ARGUMENT IS A VECTOR WHILE THE OTHER ARGUMENT IS A MATRIX,

THE RESULT WILL BE A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... | .....

THE SYMBOL | (UPSHIFT H) IS USED FOR THE MONADIC MAGNITUDE FUNCTION,
IT RETURNS THE ABSOLUTE VALUE OF NUMERIC ARGUMENTS OF ANY RANK,
FOR EXAMPLE: |0 => 0 |(¯3 ¯2 10) => 3 2 10

..... \ .....

THE SYMBOL \ (UPSHIFT /) IS USED FOLLOWING A DYADIC FUNCTION SYMBOL,
PRECEDING A SINGLE ARGUMENT PRODUCING A MIXED FUNCTION CALLED SCAN.
SCAN MAY BE USED WITH NUMERIC ARGUMENTS OF ANY RANK.
SCAN MAY ALSO BE USED WITH CHARACTER ARGUMENTS IF THE FUNCTION IT IS
COMBINED WITH ACCEPTS CHARACTER ARGUMENTS.

USING \ WITH A SCALAR SIMPLY RETURNS THE SCALAR.

WHEN USED WITH A VECTOR, SCAN GIVES THE SAME EFFECT AS INSERTING
THE OTHER FUNCTION SYMBOL BETWEEN EACH VALUE OF THE VECTOR,
THEN OPERATING PROGRESSIVELY AS FOLLOWS:
THE FIRST ELEMENT OF THE ANSWER WILL BE THE FIRST ELEMENT OF THE
ARGUMENT. THE SECOND ELEMENT WILL BE THE RESULT OF THE MAIN
FUNCTION OPERATING ON THE FIRST TWO ELEMENTS OF THE ARGUMENT.
THE THIRD ELEMENT OF THE ANSWER WILL BE THE RESULT OF THE MAIN
FUNCTION OPERATING AS IN REDUCTION ON THE FIRST THREE ELEMENTS
OF THE ARGUMENT, AND SO ON.
THE RESULT WILL ALWAYS BE THE SAME LENGTH AS THE ARGUMENT.

FOR EXAMPLE:

```
+\(1 2 3)  ⇒  1 3 6   BECAUSE  1 ⇒ 1, 1+2 ⇒ 3, 1+2+3 ⇒ 6
×\(2 3 4)  ⇒  2 6 24
```

WHEN USING SCAN, ALWAYS REMEMBER APL READS RIGHT TO LEFT WHEN IT IS
USING REDUCTION ON MORE THAN TWO ELEMENTS OF THE ARGUMENT.

```
-\(1 2 3)  ⇒  1 ¯1 2  BECAUSE  1 ⇒ 1, 1-2 ⇒ ¯1, 1-(2-3)  ⇒  2
```

WHEN USED WITH A MATRIX, SCAN WILL PERFORM AS IF EACH COLUMN IS
A UNIT. THE FIRST COLUMN OF THE ANSWER IS THE FIRST COLUMN OF
THE ARGUMENT. THE SECOND COLUMN OF THE ANSWER IS THE RESULT OF
OPERATING ON THE FIRST TWO COLUMNS OF THE ARGUMENT, AND SO ON.
APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE COLUMNS.

SINCE COLUMNS ARE THE SECOND OF TWO DIMENSIONS OF A MATRIX, THIS MEANS THAT THE SYMBOL \ WORKS 'OVER' THE LAST DIMENSION. TO SCAN OVER THE FIRST DIMENSION (E.G. ROWS OF A MATRIX), USE THE SYMBOL ⍀ (ALT .), IN PLACE OF \.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS. FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND COLUMNS, \ WILL SCAN OVER THE COLUMNS, ⍀ OVER THE PAGES. TO SCAN OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN BRACKETS FOLLOWING THE \ (OR THE ⍀) SYMBOL.

FOR EXAMPLE, TO PLUS SCAN OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY CALLED AA, ENTER +\[2]AA OR +⍀[2]AA. TO PLUS SCAN OVER THE PAGES, ENTER +⍀AA OR +\[1]AA. TO PLUS SCAN OVER THE COLUMNS, ENTER +\AA OR +⍀[3]AA.

THE RESULT WILL ALWAYS BE THE SAME SHAPE AS THE ARGUMENT.

..... ⍀ .....

THE SYMBOL ⍀ (ALT ,) IS USED FOLLOWING A DYADIC FUNCTION SYMBOL AND PRECEDING A SINGLE ARGUMENT PRODUCING A MIXED FUNCTION CALLED SCAN.

SCAN MAY BE USED WITH NUMERIC ARGUMENTS OF ANY RANK.

SCAN MAY ALSO BE USED WITH CHARACTER ARGUMENTS IF THE FUNCTION IT IS COMBINED WITH ACCEPTS CHARACTER ARGUMENTS.

USING ⍀ WITH A SCALAR SIMPLY RETURNS THE SCALAR.

WHEN USED WITH A VECTOR, SCAN GIVES THE SAME EFFECT AS INSERTING THE OTHER FUNCTION SYMBOL BETWEEN EACH VALUE OF THE VECTOR, THEN OPERATING PROGRESSIVELY AS FOLLOWS:

THE FIRST ELEMENT OF THE ANSWER WILL BE THE FIRST ELEMENT OF THE ARGUMENT. THE SECOND ELEMENT WILL BE THE RESULT OF THE MAIN FUNCTION OPERATING ON THE FIRST TWO ELEMENTS OF THE ARGUMENT, THE THIRD ELEMENT OF THE ANSWER WILL BE THE RESULT OF THE MAIN FUNCTION OPERATING AS IN REDUCTION ON THE FIRST THREE ELEMENTS OF THE ARGUMENT, AND SO ON.

THE RESULT WILL ALWAYS BE THE SAME LENGTH AS THE ARGUMENT.

FOR EXAMPLE:

```
   +⍀(1 2 3)  ⇒  1 3 6   BECAUSE  1 ⇒ 1, 1+2 ⇒ 3, 1+2+3 ⇒ 6
   ×⍀(2 3 4)  ⇒  2 6 24
```

WHEN USING SCAN, ALWAYS REMEMBER APL READS RIGHT TO LEFT WHEN IT IS USING REDUCTION ON MORE THAN TWO ELEMENTS OF THE ARGUMENT.

```
   -⍀(1 2 3)  ⇒  1 ¯1 2  BECAUSE  1 ⇒ 1, 1-2 ⇒ ¯1, 1-(2-3)  ⇒  2
```

WHEN USED WITH A MATRIX, SCAN WILL PERFORM AS IF EACH ROW IS A UNIT. THE FIRST ROW OF THE ANSWER IS THE FIRST ROW OF THE ARGUMENT. THE SECOND ROW OF THE ANSWER IS THE RESULT OF OPERATING ON THE FIRST TWO ROWS OF THE ARGUMENT, AND SO ON.

APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE ROWS.

SINCE ROWS ARE THE FIRST OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL ⌿ WORKS 'OVER' THE FIRST DIMENSION.
TO SCAN OVER THE LAST DIMENSION (E.G. COLUMNS OF A MATRIX), USE
THE SYMBOL \ (UPSHIFT /), IN PLACE OF ⌿.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS.
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND
COLUMNS, \ WILL SCAN OVER THE COLUMNS, ⌿ OVER THE PAGES.
TO SCAN OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN
BRACKETS FOLLOWING THE ⌿ (OR THE \) SYMBOL.
FOR EXAMPLE, TO PLUS SCAN OVER THE ROWS OF A THREE-DIMENSIONAL
ARRAY CALLED AA, ENTER +⌿[2]AA OR +\[2]AA. TO PLUS SCAN OVER
THE PAGES, ENTER +⌿AA OR +\[1]AA. TO PLUS SCAN OVER THE COLUMNS,
ENTER +\AA OR +⌿[3]AA.
THE RESULT WILL ALWAYS BE THE SAME SHAPE AS THE ARGUMENT.

..... ⌈ .....

THE SYMBOL ⌈ (UPSHIFT 5) IS USED FOR THE MONADIC CEILING FUNCTION,
IT RETURNS THE VALUE OF NUMERIC ARGUMENTS OF ANY RANK, 'ROUNDED UP'
TO THE NEAREST INTEGER.

FOR EXAMPLE: ⌈(¯3.4 0 1.3 2) => ¯3 0 2 2

..... ⌊ .....

THE SYMBOL ⌊ (UPSHIFT D) IS USED FOR THE MONADIC FLOOR FUNCTION,

IT RETURNS THE VALUE OF NUMERIC ARGUMENTS OF ANY RANK, 'ROUNDED DOWN'

TO THE NEAREST INTEGER,

FOR EXAMPLE: ⌊(¯3.4 0 1.3 2) => ¯4 0 1 2

..... ⌈ .....

THE SYMBOL ⌈ (UPSHIFT S) IS USED FOR THE DYADIC MAXIMUM FUNCTION,
THIS RETURNS THE MAXIMUM OF ITS TWO ARGUMENTS, WHICH ARE NUMERIC
OF ANY RANK,

FOR EXAMPLE: 2⌈3 => 3

TO CHECK A SCALAR AGAINST EACH ELEMENT OF A VECTOR, ENTER S⌈V OR V⌈S,
V1⌈V2 WILL COMPARE TWO VECTORS ELEMENT BY ELEMENT, FOR EXAMPLE:

4⌈(3 4 5) => 4 4 5 (3 4 5)⌈5 => 5 5 5

(3 4 5)⌈(1 5 9) => 3 5 9

IF THE TWO VECTORS BEING COMPARED ARE NOT THE SAME LENGTH,
A 'LENGTH ERROR' WILL RESULT,

MATRIX COMPARISON IS DONE AS FOLLOWS:
S⌈M OR M⌈S WILL CHECK THE SCALAR S AGAINST EACH ELEMENT OF
THE MATRIX M,

FOR EXAMPLE: IF M => 1 2 THEN 3⌈M => 3 3 AND M⌈3 => 3 3
3 4 3 4 3 4

M1⌈M2 WILL COMPARE THE MATRIXES M1 AND M2, ELEMENT BY ELEMENT,

FOR EXAMPLE: IF M1 => 1 2 AND M2 => 3 4 THEN M1⌈M2 => 3 4
3 4 5 6 5 6

M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT,
TRYING TO COMPARE A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR',

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS,

..... ⌊ .....

THE SYMBOL ⌊ (UPSHIFT D) IS USED FOR THE DYADIC MINIMUM FUNCTION.
THIS RETURNS THE MINIMUM OF ITS TWO ARGUMENTS, WHICH ARE NUMERIC
OF ANY RANK.

FOR EXAMPLE: 2⌊3 ⇒ 2

TO CHECK A SCALAR AGAINST EACH ELEMENT OF A VECTOR, ENTER S⌊V OR V⌊S.
V1⌊V2 WILL COMPARE TWO VECTORS ELEMENT BY ELEMENT. FOR EXAMPLE:

4⌊(3 4 5) ⇒ 3 4 4 (3 4 5)⌊5 ⇒ 3 4 5

(3 4 5)⌊(1 5 9) ⇒ 1 4 5

IF THE TWO VECTORS BEING COMPARED ARE NOT THE SAME LENGTH,
A 'LENGTH ERROR' WILL RESULT.

MATRIX COMPARISON IS DONE AS FOLLOWS:
S⌊M OR M⌊S WILL CHECK THE SCALAR S AGAINST EACH ELEMENT OF
THE MATRIX M.

FOR EXAMPLE: IF M ⇒ 1 2 THEN 3⌊M ⇒ 1 2 AND M⌊3 ⇒ 1 2
3 4 3 3 3 3

M1⌊M2 WILL COMPARE THE MATRIXES M1 AND M2, ELEMENT BY ELEMENT.

FOR EXAMPLE: IF M1 ⇒ 1 2 AND M2 ⇒ 3 4 THEN M1⌊M2 ⇒ 1 2
3 4 5 6 3 4

M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT.
TRYING TO COMPARE A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... | .....

THE SYMBOL | (UPSHIFT M) IS USED FOR THE DYADIC RESIDUE FUNCTION,
IT TAKES NUMERIC ARGUMENTS OF ANY RANK,
THE RESULT OF L|R IS THE REMAINDER WHEN R IS DIVIDED BY L, THIS IS
SIMILAR TO THE 'MODULO' FUNCTION IN OTHER COMPUTER LANGUAGES,

FOR EXAMPLE:

```
1|1.24   =>   .24        10|¯4   =>   2        ¯4|10   =>   ¯2
```

THE SIGN OF THE RESULT IS ALWAYS THE SAME AS THE SIGN OF THE LEFT
ARGUMENT, IF THE SIGNS OF THE ARGUMENTS DO NOT AGREE, THE RESULT
IS OBTAINED BY ADDING THE SMALLER ARGUMENT TO THE LARGER ONE
REPEATEDLY UNTIL THE ABSOLUTE VALUE OF THE RESULT IS LESS THAN
THE ABSOLUTE VALUE OF THE SMALLER ARGUMENT,

WHEN A SCALAR ARGUMENT IS PAIRED WITH A VECTOR ARGUMENT, THE FUNCTION
COMBINES THE SCALAR WITH EACH ELEMENT OF THE VECTOR IN TURN,

```
2|(1 2 3 4 5)   =>   1 0 1 0 1     (1 2 3 4 5)|3   =>   0 1 0 3 3
```

CORRESPONDING ELEMENTS OF TWO VECTOR ARGUMENTS ARE COMBINED,

```
(1 1.5 2)|(1 2 3)   =>   0 0.5 1
```

V_1 AND V_2 MUST HAVE THE SAME NUMBER OF ELEMENTS OR A 'LENGTH ERROR'
WILL RESULT,

WHEN A SCALAR ARGUMENT IS PAIRED WITH A MATRIX ARGUMENT, THE FUNCTION
COMBINES THE SCALAR WITH EACH ELEMENT OF THE MATRIX IN TURN,

```
FOR EXAMPLE: IF M => 1 2  THEN 3|M  =>  1 2  AND M|3  => 0 1
                    3 4                 0 1             0 3
```

CORRESPONDING ELEMENTS OF TWO MATRIX ARGUMENTS ARE COMBINED,

```
FOR EXAMPLE: IF  M1 => 1 2  AND  M2 => 3 4  THEN M1|M2  =>  2  2
                       3 4             5 6                  2  2
```

IF M_1 AND M_2 ARE NOT THE SAME SHAPE, A 'LENGTH ERROR' WILL OCCUR,
IF ONE ARGUMENT IS A VECTOR WHILE THE OTHER ARGUMENT IS A MATRIX,

THE RESULT WILL BE A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... ⍋ .....

THE SYMBOL ⍋ (ALT 4) IS USED FOR THE MONADIC GRADE UP FUNCTION.
IT TAKES A NUMERIC VECTOR ARGUMENT AND RETURNS A VECTOR OF THE
INDEXES OF THE ELEMENTS IN ORDER FROM SMALLEST TO LARGEST.
IN OTHER WORDS, THE FIRST ELEMENT IN THE RESPONSE IS THE INDEX
OF THE SMALLEST ELEMENT IN THE INPUT. THE LAST ELEMENT IN THE
RESPONSE IS THE INDEX OF THE LARGEST ELEMENT IN THE INPUT.
EQUAL VALUES WILL BE GRADED IN ORDER FROM LEFT TO RIGHT.

FOR EXAMPLE: ⍋ 4 8 6 3 ⍝ 4 1 3 2
BECAUSE THE FOURTH ELEMENT OF THE INPUT (3) IS THE SMALLEST, ETC.
THIS CAN BE USED TO SORT A VECTOR BY COMBINING WITH INDEX BRACKETS:
IF V ⍝ 4 8 6 3 THEN V[⍋V] ⍝ 3 4 6 8

ON THE OTHER HAND, ⍋⍋V WILL PRODUCE A VECTOR OF THE POSITION NUMBERS
(RANK ORDER) CORRESPONDING TO THE INPUT VECTOR.
FOR EXAMPLE: ⍋⍋ 4 8 6 3 ⍝ 2 4 3 1
SINCE 4 IS THE SECOND SMALLEST ELEMENT, 8 IS THE FOURTH SMALLEST, ETC.

..... ⍒ .....

THE SYMBOL ⍒ (ALT 3) IS USED FOR THE MONADIC GRADE DOWN FUNCTION.
IT TAKES A NUMERIC VECTOR ARGUMENT AND RETURNS A VECTOR OF THE
INDEXES OF THE ELEMENTS IN ORDER FROM LARGEST TO SMALLEST.
IN OTHER WORDS, THE FIRST ELEMENT IN THE RESPONSE IS THE INDEX
OF THE LARGEST ELEMENT IN THE INPUT. THE LAST ELEMENT IN THE
RESPONSE IS THE INDEX OF THE SMALLEST ELEMENT IN THE INPUT.
EQUAL VALUES WILL BE GRADED IN ORDER FROM LEFT TO RIGHT.

FOR EXAMPLE: ⍒ 4 8 6 3 => 2 3 1 4
BECAUSE THE SECOND ELEMENT OF THE INPUT (8) IS THE LARGEST, ETC.
THIS CAN BE USED TO SORT A VECTOR BY COMBINING WITH INDEX BRACKETS:
IF V => 4 8 6 3 THEN V[⍒V] => 8 6 4 3

ON THE OTHER HAND, ⍒⍒V WILL PRODUCE A VECTOR OF THE POSITION NUMBERS
(IN REVERSE ORDER) CORRESPONDING TO THE INPUT VECTOR.
FOR EXAMPLE: ⍒⍒ 4 8 6 3 => 3 1 2 4
SINCE 4 IS THE THIRD LARGEST ELEMENT, 8 IS THE LARGEST, ETC.

..... ? .....

THE SYMBOL ? (UPSHIFT Q) IS USED FOR THE MONADIC ROLL FUNCTION.

THIS IS A RANDOM NUMBER GENERATOR WHICH TAKES POSITIVE INTEGER ARGUMENTS AND RETURNS A RANDOMLY SELECTED INTEGER UP TO THE VALUE OF THE ARGUMENT.

THE LOWEST POSSIBLE VALUE WHICH CAN BE RETURNED IS NORMALLY (BY DEFAULT) 1.

USING ? WITH NEGATIVE ARGUMENTS OR ZERO PRODUCES A 'DOMAIN ERROR'.

FOR EXAMPLE: ?3 => EITHER 1, 2, OR 3, WITH EQUAL PROBABILITY

WHEN USED WITH VECTORS, ROLL RETURNS A VECTOR WITH EACH NUMBER SELECTED RANDOMLY FROM THE RANGE DETERMINED BY THE CORRESPONDING VALUE IN THE ARGUMENT.

FOR EXAMPLE:

?3 3 3 => 1 1 2 OR 1 3 2 OR 3 3 2 OR 2 1 3 ETC.

?6 6 SIMULATES THE ROLLING OF A PAIR OF DICE.

ROLL CAN BE USED WITH MATRIXES OR HIGHER LEVEL ARRAYS IN A SIMILAR FASHION.

..... ? .....

THE SYMBOL ? (UPSHIFT Q) IS USED FOR THE DYADIC DEAL FUNCTION.

THE RESULT OF THIS FUNCTION IS AN ARRAY OF RANDOM NUMBERS.

THE SHAPE OF THE ARRAY IS DETERMINED BY THE LEFT ARGUMENT.

THE NUMBERS ARE RANDOMLY SELECTED FROM THE POSITIVE INTEGERS

UP TO THE VALUE OF THE RIGHT ARGUMENT, WITH NO REPLACEMENT.

THAT IS, A NUMBER CANNOT APPEAR TWICE IN THE RESULT.

BOTH ARGUMENTS MUST BE A POSITIVE SINGLE NUMBER (SCALAR OR VECTOR

OF LENGTH 1), AND THE RIGHT MUST EQUAL OR EXCEED THE LEFT.

USING HIGHER-ORDER ARGUMENTS PRODUCES A 'RANK ERROR'.

USING NEGATIVE ARGUMENTS OR ZERO PRODUCES A 'DOMAIN ERROR'.

FOR EXAMPLE: 2?2 => 1 2 OR 2 1 BUT NOT 2 2 OR 1 1

5?52 SIMULATES THE DEAL OF A HAND OF CARDS.

..... = ...

THE SYMBOL = (UPSHIFT 5) IS USED FOR THE DYADIC EQUALS FUNCTION,
THIS COMPARES NUMERIC OR CHARACTER ARGUMENTS OF ANY RANK,
IT RETURNS 1 FOR EACH ELEMENT OF THE LEFT ARGUMENT THAT IS IDENTICAL
TO THE CORRESPONDING ELEMENT OF THE RIGHT ARGUMENT, AND 0 FOR
EACH ELEMENT THAT IS NOT,

FOR EXAMPLE: 2=2 => 1 2=4 => 0 'A'='B' => 0

TO COMPARE A SCALAR TO EACH ELEMENT OF A VECTOR,
ENTER EITHER S=V OR V=S,
2=(2 3 4) => 1 0 0 (3 4 5)=5 => 0 0 1
V1=V2 WILL COMPARE TWO VECTORS ELEMENT BY ELEMENT, FOR EXAMPLE:
'THINK'='THANK' => 1 1 0 1 1
IF THE TWO VECTORS BEING COMPARED ARE NOT THE SAME LENGTH,
YOU WILL GET A 'LENGTH ERROR',

MATRIX COMPARISON IS DONE AS FOLLOWS:
S=M OR M=S WILL COMPARE THE SCALAR S TO EACH ELEMENT OF THE MATRIX M,
FOR EXAMPLE: IF M => 1 2 THEN 3=M => 0 0 AND M=3 => 0 0
3 4 1 0 1 0
M1=M2 WILL COMPARE THE MATRIX M1 TO THE MATRIX M2, ELEMENT BY ELEMENT,
FOR EXAMPLE: IF M1 => HOW AND M2 => OOH THEN M1=M2 => 0 1 0
NOW WOW 0 1 1
M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT,
COMPARING A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR',

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS,

..... ≠ .....

THE SYMBOL ≠ (UPSHIFT 8) IS USED FOR THE DYADIC NOT EQUAL FUNCTION.
THIS COMPARES NUMERIC OR CHARACTER ARGUMENTS OF ANY RANK.
IT RETURNS 1 FOR EACH ELEMENT OF THE LEFT ARGUMENT THAT IS NOT EQUAL
TO THE CORRESPONDING ELEMENT OF THE RIGHT ARGUMENT, AND 0 FOR
EACH ELEMENT THAT IS.

FOR EXAMPLE: 2≠2 ⇒ 0 2≠4 ⇒ 1 'A'≠'B' ⇒ 1

TO COMPARE A SCALAR TO EACH ELEMENT OF A VECTOR,
ENTER EITHER S≠V OR V≠S.
2≠(2 3 4) ⇒ 0 1 1 (3 4 5)≠5 ⇒ 1 1 0
V1≠V2 WILL COMPARE TWO VECTORS ELEMENT BY ELEMENT. FOR EXAMPLE:
'THINK'≠'THANK' ⇒ 0 0 1 0 0
IF THE TWO VECTORS BEING COMPARED ARE NOT THE SAME LENGTH,
YOU WILL GET A 'LENGTH ERROR'.

MATRIX COMPARISON IS DONE AS FOLLOWS:
S≠M OR M≠S WILL COMPARE THE SCALAR S TO EACH ELEMENT OF THE MATRIX M.
FOR EXAMPLE: IF M ⇒ 1 2 THEN 3≠M ⇒ 1 1 AND M≠3 ⇒ 1 1
3 4 0 1 0 1
M1≠M2 WILL COMPARE THE MATRIX M1 TO THE MATRIX M2, ELEMENT BY ELEMENT.
FOR EXAMPLE: IF M1 ⇒ HOW AND M2 ⇒ OOH THEN M1≠M2 ⇒ 1 0 1
NOW WOW 1 0 0
M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT.
COMPARING A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... ⍝ .....

THE SYMBOL ⍝ (UPSHIFT 3) IS USED FOR THE DYADIC LESS THAN FUNCTION.
THIS COMPARES NUMERIC ARGUMENTS OF ANY RANK.
IT RETURNS 1 FOR EACH ELEMENT OF THE LEFT ARGUMENT THAT IS LESS
THAN THE CORRESPONDING ELEMENT OF THE RIGHT ARGUMENT, AND 0 FOR
EACH ELEMENT THAT IS NOT.

FOR EXAMPLE: 2⍝2 => 0 2⍝4 => 1

TO COMPARE A SCALAR TO EACH ELEMENT OF A VECTOR,
ENTER EITHER S⍝V OR V⍝S. ORDER IS IMPORTANT.
2⍝(1 2 3) => 0 0 1 (3 4 5)⍝5 => 1 1 0
V1⍝V2 WILL COMPARE TWO VECTORS ELEMENT BY ELEMENT. FOR EXAMPLE:
(1 2 3)⍝(1 3 5) => 0 1 1
IF THE TWO VECTORS BEING COMPARED ARE NOT THE SAME LENGTH,
YOU WILL GET A 'LENGTH ERROR'.

MATRIX COMPARISON IS DONE AS FOLLOWS:
S⍝M OR M⍝S WILL COMPARE THE SCALAR S TO EACH ELEMENT OF THE MATRIX M.
FOR EXAMPLE: IF M => 1 2 THEN 3⍝M => 0 0 AND M⍝3 => 1 1
3 4 0 1 0 0
M1⍝M2 WILL COMPARE THE MATRIX M1 TO THE MATRIX M2, ELEMENT BY ELEMENT.
FOR EXAMPLE: IF M1 => 1 4 AND M2 => 2 6 THEN M1⍝M2 => 1 1
4 7 5 3 1 0
M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT.
COMPARING A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... ≤ .....

THE SYMBOL ≤ (UPSHIFT 4) IS USED FOR THE DYADIC LESS THAN OR EQUAL TO
FUNCTION. THIS COMPARES NUMERIC ARGUMENTS OF ANY RANK.
IT RETURNS 1 FOR EACH ELEMENT OF THE LEFT ARGUMENT THAT IS LESS THAN
OR EQUAL TO THE CORRESPONDING ELEMENT OF THE RIGHT ARGUMENT, AND
0 FOR EACH ELEMENT THAT IS NOT.

FOR EXAMPLE: 2≤2 ⇒ 1 2≤4 ⇒ 1 2≤¯2 ⇒ 0

TO COMPARE A SCALAR TO EACH ELEMENT OF A VECTOR,
ENTER EITHER S≤V OR V≤S. ORDER IS IMPORTANT.
2≤(1 2 3) ⇒ 0 1 1 (3 4 5)≤5 ⇒ 1 1 1
V1≤V2 WILL COMPARE TWO VECTORS ELEMENT BY ELEMENT. FOR EXAMPLE:
(1 2 3)≤(1 3 5) ⇒ 1 1 1
IF THE TWO VECTORS BEING COMPARED ARE NOT THE SAME LENGTH,
YOU WILL GET A 'LENGTH ERROR'.

MATRIX COMPARISON IS DONE AS FOLLOWS:
S≤M OR M≤S WILL COMPARE THE SCALAR S TO EACH ELEMENT OF THE MATRIX M.
FOR EXAMPLE: IF M ⇒ 1 2 THEN 3≤M ⇒ 0 0 AND M≤3 ⇒ 1 1
3 4 1 1 1 0
M1≤M2 WILL COMPARE THE MATRIX M1 TO THE MATRIX M2, ELEMENT BY ELEMENT.
FOR EXAMPLE: IF M1 ⇒ 1 4 AND M2 ⇒ 2 6 THEN M1≤M2 ⇒ 1 1
4 7 5 3 1 0
M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT.
COMPARING A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... ≥ .....

THE SYMBOL ≥ (UPSHIFT 6) IS USED FOR THE DYADIC GREATER THAN OR EQUAL TO FUNCTION. THIS COMPARES NUMERIC ARGUMENTS OF ANY RANK.

IT RETURNS 1 FOR EACH ELEMENT OF THE LEFT ARGUMENT THAT IS GREATER THAN OR EQUAL TO THE CORRESPONDING ELEMENT OF THE RIGHT ARGUMENT, AND 0 FOR EACH ELEMENT THAT IS NOT.

FOR EXAMPLE: 2≥2 ⇒ 1 2≥4 ⇒ 0 2≥¯2 ⇒ 1

TO COMPARE A SCALAR TO EACH ELEMENT OF A VECTOR, ENTER EITHER S≥V OR V≥S. ORDER IS IMPORTANT.

2≥(1 2 3) ⇒ 1 1 0 (3 4 5)≥5 ⇒ 0 0 1

V1≥V2 WILL COMPARE TWO VECTORS ELEMENT BY ELEMENT. FOR EXAMPLE:

(1 2 3)≥(1 3 5) ⇒ 1 0 0

IF THE TWO VECTORS BEING COMPARED ARE NOT THE SAME LENGTH, YOU WILL GET A 'LENGTH ERROR'.

MATRIX COMPARISON IS DONE AS FOLLOWS:

S≥M OR M≥S WILL COMPARE THE SCALAR S TO EACH ELEMENT OF THE MATRIX M.

FOR EXAMPLE: IF M ⇒ 1 2 THEN 3≥M ⇒ 1 1 AND M≥3 ⇒ 0 0
3 4 1 0 1 1

M1≥M2 WILL COMPARE THE MATRIX M1 TO THE MATRIX M2, ELEMENT BY ELEMENT.

FOR EXAMPLE: IF M1 ⇒ 1 4 AND M2 ⇒ 2 6 THEN M1≥M2 ⇒ 0 0
4 7 5 3 0 1

M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT.

COMPARING A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR'.

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

..... ⟩ .....

THE SYMBOL ⟩ (UPSHIFT 7) IS USED FOR THE DYADIC GREATER THAN
FUNCTION, THIS COMPARES NUMERIC ARGUMENTS OF ANY RANK,
IT RETURNS 1 FOR EACH ELEMENT OF THE LEFT ARGUMENT THAT IS GREATER
THAN THE CORRESPONDING ELEMENT OF THE RIGHT ARGUMENT, AND 0 FOR
EACH ELEMENT THAT IS NOT,

FOR EXAMPLE: 2⟩2 ⟹ 0 4⟩2 ⟹ 1

TO COMPARE A SCALAR TO EACH ELEMENT OF A VECTOR,
ENTER EITHER S⟩V OR V⟩S, ORDER IS IMPORTANT,
2⟩(1 2 3) ⟹ 1 0 0 (3 4 5)⟩5 ⟹ 0 0 0
V1⟩V2 WILL COMPARE TWO VECTORS ELEMENT BY ELEMENT, FOR EXAMPLE:
(1 2 3)⟩(1 3 5) ⟹ 0 0 0
IF THE TWO VECTORS BEING COMPARED ARE NOT THE SAME LENGTH,
YOU WILL GET A 'LENGTH ERROR',

MATRIX COMPARISON IS DONE AS FOLLOWS:
S⟩M OR M⟩S WILL COMPARE THE SCALAR S TO EACH ELEMENT OF THE MATRIX M,
FOR EXAMPLE: IF M ⟹ 1 2 THEN 3⟩M ⟹ 1 1 AND M⟩3 ⟹ 0 0
3 4 0 0 0 1
M1⟩M2 WILL COMPARE THE MATRIX M1 TO THE MATRIX M2, ELEMENT BY ELEMENT,
FOR EXAMPLE: IF M1 ⟹ 1 4 AND M2 ⟹ 2 6 THEN M1⟩M2 ⟹ 0 0
4 7 1 3 1 1
M1 AND M2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT,
COMPARING A VECTOR TO A MATRIX WILL RESULT IN A 'RANK ERROR',

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS,

..... ~ .....

THE SYMBOL ~ (UPSHIFT T) IS USED FOR THE MONADIC NOT FUNCTION.
IT TAKES AN ARGUMENT OF ANY RANK WHICH CONSISTS ONLY OF 1'S AND 0'S,
AND RETURNS THE LOGICAL INVERSE. THAT IS, 1'S ARE CHANGED TO 0'S
AND 0'S ARE CHANGED TO 1'S.

FOR EXAMPLE: ~1 0 0 1 => 0 1 1 0

..... ^ .....

THE SYMBOL ^ (UPSHIFT 6) IS USED FOR THE DYADIC AND FUNCTION.

IT TAKES ARGUMENTS OF ANY RANK WHICH CONSIST ONLY OF 1'S AND 0'S. THIS FUNCTION RETURNS 1 WHERE BOTH ARGUMENTS ARE 1, AND 0 OTHERWISE. THE ORDER OF THE ARGUMENTS IS NOT IMPORTANT, BUT ENTERING AN ARGUMENT WHICH IS NOT 1 OR 0 WILL PRODUCE A 'DOMAIN ERROR'.

FOR EXAMPLE: 1^1 => 1 1^0 => 0 0^0 => 0

TO COMPARE A SCALAR TO EACH ELEMENT OF AN ARRAY,

ENTER EITHER S^A OR A^S. FOR EXAMPLE, IF V => 2 ¯3 0

1^(×V) => 1 0 0 (V=2) ^ V[1]=2 => 1 0 0

A1^A2 WILL COMPARE TWO ARRAYS ELEMENT BY ELEMENT. FOR EXAMPLE:

(V>¯3) ^ (V<2) => 0 0 1

IF M1 => 1 0 AND M2 => 1 1 THEN M1^M2 => 1 0
1 0 0 0 0 0

A1 AND A2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT. COMPARING TWO ARRAYS OF DIFFERENT RANK, FOR EXAMPLE A VECTOR AND A MATRIX, WILL RESULT IN A 'RANK ERROR'.

..... * .....

THE SYMBOL * (ALT 0) IS USED FOR THE DYADIC NOT AND FUNCTION.
IT TAKES ARGUMENTS OF ANY RANK WHICH CONSIST ONLY OF 1'S AND 0'S.
THIS FUNCTION RETURNS 1 WHERE ONE OR NEITHER OF THE ARGUMENTS ARE 1,
AND 0 OTHERWISE (WHERE BOTH ARE 1).
THE ORDER OF THE ARGUMENTS IS NOT IMPORTANT, BUT ENTERING AN ARGUMENT
WHICH IS NOT 1 OR 0 WILL PRODUCE A 'DOMAIN ERROR'.

FOR EXAMPLE: 1*1 => 0 1*0 => 1 0*0 => 1

TO COMPARE A SCALAR TO EACH ELEMENT OF AN ARRAY,
ENTER EITHER S*A OR A*S. FOR EXAMPLE, IF V => 2 ¯3 0
1*(×V) => 0 1 1 (V=2) * V[1]=2 => 0 1 1
A1*A2 WILL COMPARE TWO ARRAYS ELEMENT BY ELEMENT. FOR EXAMPLE:
(V)¯3) * (V(2) => 1 1 0
IF M1 => 1 0 AND M2 => 1 1 THEN M1*M2 => 0 1
1 0 0 0 1 1

A1 AND A2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT.
COMPARING TWO ARRAYS OF DIFFERENT RANK, FOR EXAMPLE A VECTOR AND A
MATRIX, WILL RESULT IN A 'RANK ERROR'.

..... ∨ .....

THE SYMBOL ∨ (UPSHIFT 9) IS USED FOR THE DYADIC OR FUNCTION,
IT TAKES ARGUMENTS OF ANY RANK WHICH CONSIST ONLY OF 1'S AND 0'S,
THIS FUNCTION RETURNS 1 WHERE ONE OR BOTH ARGUMENTS ARE 1,
AND 0 OTHERWISE (WHERE BOTH ARE 0),
THE ORDER OF THE ARGUMENTS IS NOT IMPORTANT, BUT ENTERING AN ARGUMENT
WHICH IS NOT 1 OR 0 WILL PRODUCE A 'DOMAIN ERROR',

FOR EXAMPLE; 1∨1 ⇒ 1 1∨0 ⇒ 1 0∨0 ⇒ 0

TO COMPARE A SCALAR TO EACH ELEMENT OF AN ARRAY,
ENTER EITHER S∨A OR A∨S, FOR EXAMPLE, IF V ⇒ 2 ¯3 0
1∨(×V) ⇒ 1 1 1 (V=2) ∨ V[1]=3 ⇒ 1 0 0
$A_1 \vee A_2$ WILL COMPARE TWO ARRAYS ELEMENT BY ELEMENT, FOR EXAMPLE;
(V>¯3) ∨ (V<2) ⇒ 1 1 1
IF M_1 ⇒ 1 0 AND M_2 ⇒ 1 1 THEN $M_1 \vee M_2$ ⇒ 1 1
1 0 0 0 1 0

A_1 AND A_2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT,
COMPARING TWO ARRAYS OF DIFFERENT RANK, FOR EXAMPLE A VECTOR AND A
MATRIX, WILL RESULT IN A 'RANK ERROR',

..... ⍱ .....

THE SYMBOL ⍱ (ALT 9) IS USED FOR THE DYADIC NOR FUNCTION.

IT TAKES ARGUMENTS OF ANY RANK WHICH CONSIST ONLY OF 1'S AND 0'S.

THIS FUNCTION RETURNS 1 WHERE NEITHER OF THE ARGUMENTS ARE 1,

AND 0 OTHERWISE.

THE ORDER OF THE ARGUMENTS IS NOT IMPORTANT, BUT ENTERING AN ARGUMENT

WHICH IS NOT 1 OR 0 WILL PRODUCE A 'DOMAIN ERROR'.

FOR EXAMPLE: 1⍱1 ⍝ 0 1⍱0 ⍝ 0 0⍱0 ⍝ 1

TO COMPARE A SCALAR TO EACH ELEMENT OF AN ARRAY,

ENTER EITHER S⍱A OR A⍱S. FOR EXAMPLE, IF V ⍝ 2 ¯3 0

1⍱(×V) ⍝ 0 0 0 (V=2) ⍱ V[1]=3 ⍝ 0 1 1

A1⍱A2 WILL COMPARE TWO ARRAYS ELEMENT BY ELEMENT. FOR EXAMPLE:

(V>¯3) ⍱ (V<2) ⍝ 0 0 0

IF M1 ⍝ 1 0 AND M2 ⍝ 1 1 THEN M1⍱M2 ⍝ 0 0

1 0 0 0 0 1

A1 AND A2 MUST BE THE SAME SHAPE OR A 'LENGTH ERROR' WILL RESULT.

COMPARING TWO ARRAYS OF DIFFERENT RANK, FOR EXAMPLE A VECTOR AND A

MATRIX, WILL RESULT IN A 'RANK ERROR'.

..... ε .....

THE SYMBOL ε (UPSHIFT E) IS USED FOR THE DYADIC MEMBER OF FUNCTION.
THIS COMPARES NUMERIC OR CHARACTER ARGUMENTS OF ANY RANK.
IT RETURNS 1 FOR EACH ELEMENT OF THE LEFT ARGUMENT THAT IS FOUND
ANYWHERE IN THE RIGHT ARGUMENT, AND 0 FOR EVERY ELEMENT
THAT IS NOT.

WITH TWO SCALAR ARGUMENTS, ε IS EQUIVALENT TO = .
FOR EXAMPLE: 2ε2 ⇒ 1 2ε4 ⇒ 0 'A'ε'B' ⇒ 0

TO DETERMINE IF A SCALAR IS A MEMBER OF AN ARRAY, ENTER SεA.
2ε(2 3 4) ⇒ 1 'A'ε'ABC' ⇒ 1
ENTERING AεS WILL PRODUCE THE SAME RESULT AS A=S.

A1εA2 WILL TAKE EACH ELEMENT OF THE LEFT ARRAY AND DETERMINE IF
IT IS CONTAINED IN THE RIGHT ARRAY.
FOR EXAMPLE: 'WHICH 1'ε'0123456789' ⇒ 0 0 0 0 0 0 1
IF A1 ⇒ 1 22 14 AND A2 ⇒ 1 2 3
38 5 2
THEN A1εA2 ⇒ 1 0 0 AND A2εA1 ⇒ 1 1 0
0 0 1
THE RESULT OF A1εA2 HAS THE SAME SHAPE AS A1.

..... ⌽ .....

THE SYMBOL ⌽ (ALT 5) IS USED FOR THE MONADIC REVERSE FUNCTION,
IT TAKES NUMERIC OR CHARACTER ARGUMENTS OF ANY RANK,
IT RETURNS ALL THE ELEMENTS OF THE ARGUMENT IN THE SAME SHAPE,
BUT IN A DIFFERENT ORDER, AS FOLLOWS.

USING ⌽ WITH A SCALAR SIMPLY RETURNS THE SCALAR.

USING ⌽ WITH A VECTOR RETURNS THE VECTOR IN REVERSE ORDER.
FOR EXAMPLE: ⌽2 3 5 ⇒ 5 3 2
⌽'STOP' ⇒ POTS

WHEN USED WITH A MATRIX, ⌽ TAKES THE COLUMNS AS UNITS AND REVERSES
THEM. THE FIRST COLUMN BECOMES THE LAST COLUMN, AND SO ON.

```
FOR EXAMPLE, IF M ⇒  1 2 3   THEN   ⌽M   ⇒   3 2 1
                     4 5 6                    6 5 4
                     7 8 9                    9 8 7
```

APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE COLUMNS,
SINCE COLUMNS ARE THE SECOND OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL ⌽ WORKS 'OVER' THE LAST DIMENSION.
TO REVERSE OVER THE FIRST DIMENSION (E.G. ROWS OF A MATRIX), USE THE
SYMBOL ⊖ (ALT 7), IN PLACE OF ⌽.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS.
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND
COLUMNS, ⌽ WILL REVERSE OVER THE COLUMNS, ⊖ OVER THE PAGES.
TO REVERSE OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN
BRACKETS FOLLOWING THE ⌽ (OR THE ⊖) SYMBOL.
FOR EXAMPLE, TO REVERSE OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY
CALLED AA, ENTER ⌽[2]AA OR ⊖[2]AA. TO REVERSE OVER THE PAGES,
ENTER ⊖AA OR ⌽[1]AA. TO REVERSE OVER THE COLUMNS, ENTER ⌽AA
OR ⊖[3]AA.

..... ⌽ .....

THE SYMBOL ⌽ (ALT 7) IS USED FOR THE MONADIC REVERSE FUNCTION.
IT TAKES NUMERIC OR CHARACTER ARGUMENTS OF ANY RANK,
IT RETURNS ALL THE ELEMENTS OF THE ARGUMENT IN THE SAME SHAPE,
BUT IN A DIFFERENT ORDER, AS FOLLOWS.

USING ⌽ WITH A SCALAR SIMPLY RETURNS THE SCALAR.

USING ⌽ WITH A VECTOR RETURNS THE VECTOR IN REVERSE ORDER.
FOR EXAMPLE: ⌽2 3 5 => 5 3 2
⌽'STOP' => POTS

WHEN USED WITH A MATRIX, ⌽ TAKES THE ROWS AS UNITS AND REVERSES
THEM. THE FIRST ROW BECOMES THE LAST ROW, AND SO ON.
FOR EXAMPLE, IF M => 1 2 3 THEN ⌽M => 7 8 9
4 5 6 4 5 6
7 8 9 1 2 3
APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE ROWS.
SINCE ROWS ARE THE FIRST OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL ⌽ WORKS 'OVER' THE FIRST DIMENSION.
TO REVERSE OVER THE LAST DIMENSION (E.G. COLUMNS OF A MATRIX),
USE THE SYMBOL ⊖ (ALT 5), IN PLACE OF ⌽.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS.
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND
COLUMNS, ⊖ WILL REVERSE OVER THE COLUMNS, ⌽ OVER THE PAGES.
TO REVERSE OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN
BRACKETS FOLLOWING THE ⌽ (OR THE ⊖) SYMBOL.
FOR EXAMPLE, TO REVERSE OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY
CALLED AA, ENTER ⌽[2]AA OR ⊖[2]AA. TO REVERSE OVER THE PAGES,
ENTER ⌽AA OR ⊖[1]AA. TO REVERSE OVER THE COLUMNS, ENTER ⊖AA
OR ⌽[3]AA.

..... ⌽ .....

THE SYMBOL ⌽ (ALT 5) IS USED FOR THE DYADIC ROTATE FUNCTION,
THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK,
THE LEFT ARGUMENT MUST BE AN INTEGER SCALAR OR ARRAY,
IT RETURNS ALL THE ELEMENTS OF THE RIGHT ARGUMENT IN THE SAME SHAPE,
BUT IN A DIFFERENT ORDER, AS FOLLOWS,

USING ⌽ WITH TWO SCALAR ARGUMENTS SIMPLY RETURNS THE RIGHT ARGUMENT,

A VECTOR RIGHT ARGUMENT REQUIRES A SCALAR LEFT ARGUMENT,
EACH ELEMENT OF THE RIGHT ARGUMENT IS MOVED TO THE LEFT BY THE NUMBER
OF PLACES INDICATED IN THE LEFT ARGUMENT,
FOR EXAMPLE: 2⌽2 3 5 ⇒ 5 2 3 3⌽'ROTATE' ⇒ ATEROT

WHEN USED WITH A MATRIX, ⌽ TAKES EITHER A SCALAR LEFT ARGUMENT OR A
VECTOR WITH AS MANY ELEMENTS AS THERE ARE ROWS IN THE MATRIX,
S⌽M TAKES THE COLUMNS AS UNITS AND MOVES EACH COLUMN TO THE LEFT BY
THE NUMBER OF SPACES INDICATED BY THE SCALAR,
FOR EXAMPLE: IF M ⇒ 1 2 3 THEN 2⌽M ⇒ 3 1 2
4 5 6 6 4 5
7 8 9 9 7 8
V⌽M TAKES EACH ROW INDIVIDUALLY AND MOVES THE ELEMENTS TO THE LEFT BY
THE NUMBER OF SPACES INDICATED BY THE CORRESPONDING ELEMENT OF THE
LEFT ARGUMENT, FOR EXAMPLE:

2 3 1⌽M ⇒ 3 1 2 THE FIRST ROW IS MOVED 2 SPACES LEFT
4 5 6 THE SECOND ROW IS ROTATED 3 SPACES
8 9 7 THE THIRD ROW IS ROTATED ONCE

EACH ELEMENT IS IN THE SAME ROW BUT A DIFFERENT COLUMN, SO APL
USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE COLUMNS,
SINCE COLUMNS ARE THE SECOND OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL ⌽ WORKS 'OVER' THE LAST DIMENSION,
TO ROTATE OVER THE FIRST DIMENSION (E,G, ROWS OF A MATRIX), USE THE

SYMBOL ⍟ (ALT 7), IN PLACE OF ⌽.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS. FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND COLUMNS, ⌽ WILL ROTATE OVER THE COLUMNS, ⍟ OVER THE PAGES. TO ROTATE OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN BRACKETS FOLLOWING THE ⌽ (OR THE ⍟) SYMBOL.

THE LEFT ARGUMENT MUST HAVE THE SAME SHAPE AS THE RIGHT ARGUMENT OMITTING THE DIMENSION BEING ROTATED OVER, OR IT MAY BE A SCALAR. FOR EXAMPLE, TO ROTATE OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY CALLED AA WHERE ⍴AA ⇒ 2 3 4, ENTER EITHER S⌽[2]AA OR M⌽[2]AA WHERE ⍴M ⇒ 2 4. TO ROTATE OVER THE PAGES, ENTER S⍟AA OR M⍟AA OR S⌽[1]AA OR M⌽[1]AA WHERE ⍴M ⇒ 3 4. TO ROTATE OVER THE COLUMNS, ENTER S⌽AA OR M⌽AA OR S⍟[3]AA OR M⍟[3]AA WHERE ⍴M ⇒ 2 3.

..... ⌽ .....

THE SYMBOL ⌽ (ALT 7) IS USED FOR THE DYADIC ROTATE FUNCTION.

THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK.

THE LEFT ARGUMENT MUST BE AN INTEGER SCALAR OR ARRAY.

IT RETURNS ALL THE ELEMENTS OF THE RIGHT ARGUMENT IN THE SAME SHAPE,

BUT IN A DIFFERENT ORDER, AS FOLLOWS.

USING ⌽ WITH TWO SCALAR ARGUMENTS SIMPLY RETURNS THE RIGHT ARGUMENT.

A VECTOR RIGHT ARGUMENT REQUIRES A SCALAR LEFT ARGUMENT.

EACH ELEMENT OF THE RIGHT ARGUMENT IS MOVED TO THE LEFT BY THE NUMBER

OF PLACES INDICATED IN THE LEFT ARGUMENT.

FOR EXAMPLE: 2⌽2 3 5 ⟹ 5 2 3 3⌽'ROTATE' ⟹ ATEROT

WHEN USED WITH A MATRIX, ⌽ TAKES EITHER A SCALAR LEFT ARGUMENT OR A

VECTOR WITH AS MANY ELEMENTS AS THERE ARE COLUMNS IN THE MATRIX.

S⌽M TAKES THE ROWS AS UNITS AND MOVES EACH ROW UP BY THE NUMBER

SPACES INDICATED BY THE SCALAR.

FOR EXAMPLE: IF M ⟹ 1 2 3 THEN 2⌽M ⟹ 7 8 9

4 5 6 1 2 3

7 8 9 4 5 6

V⌽M TAKES EACH COLUMN INDIVIDUALLY AND MOVES THE ELEMENTS UP BY

THE NUMBER OF SPACES INDICATED BY THE CORRESPONDING ELEMENT OF THE

LEFT ARGUMENT. FOR EXAMPLE:

2 3 1⌽M ⟹ 7 2 6 THE FIRST COLUMN IS MOVED 2 SPACES UP

1 5 9 THE SECOND COLUMN IS ROTATED 3 SPACES

4 8 3 THE THIRD COLUMN IS ROTATED ONCE

EACH ELEMENT IS IN THE SAME COLUMN BUT A DIFFERENT ROW, SO APL

USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE ROWS.

SINCE ROWS ARE THE FIRST OF TWO DIMENSIONS OF A MATRIX,

THIS MEANS THAT THE SYMBOL ⌽ WORKS 'OVER' THE FIRST DIMENSION.

TO ROTATE OVER THE LAST DIMENSION (E.G. COLUMNS OF A MATRIX), USE THE

SYMBOL ⌽ (ALT 5), IN PLACE OF ⍉.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS. FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND COLUMNS, ⌽ WILL ROTATE OVER THE COLUMNS, ⍉ OVER THE PAGES. TO ROTATE OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN BRACKETS FOLLOWING THE ⍉ (OR THE ⌽) SYMBOL.

THE LEFT ARGUMENT MUST HAVE THE SAME SHAPE AS THE RIGHT ARGUMENT OMITTING THE DIMENSION BEING ROTATED OVER, OR IT MAY BE A SCALAR. FOR EXAMPLE, TO ROTATE OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY CALLED AA WHERE ⍴AA ⇒ 2 3 4, ENTER EITHER S⌽[2]AA OR M⌽[2]AA WHERE ⍴M ⇒ 2 4. TO ROTATE OVER THE PAGES, ENTER S⍉AA OR M⍉AA OR S⌽[1]AA OR M⌽[1]AA WHERE ⍴M ⇒ 3 4. TO ROTATE OVER THE COLUMNS, ENTER S⌽AA OR M⌽AA OR S⍉[3]AA OR M⍉[3]AA WHERE ⍴M ⇒ 2 3.

..... ⍉ .....

THE SYMBOL ⍉ (ALT 6) IS USED FOR THE MONADIC TRANSPOSE FUNCTION.
IT TAKES NUMERIC OR CHARACTER ARGUMENTS OF ANY RANK.
IT RETURNS ALL THE ELEMENTS OF THE ARGUMENT IN A DIFFERENT SHAPE AND ORDER, AS FOLLOWS.

USING ⍉ WITH A SCALAR OR VECTOR SIMPLY RETURNS THE ARGUMENT.

WHEN USED WITH A MATRIX, ⍉ PERFORMS MATRIX TRANSPOSITION;
THE ROWS BECOME COLUMNS AND THE COLUMNS BECOME ROWS.

```
FOR EXAMPLE, IF M => 1 2 3   THEN   ⍉M   => 1 4 7
                    4 5 6                   2 5 8
                    7 8 9                   3 6 9
```

WHEN USED WITH HIGHER-DIMENSIONAL ARRAYS, TRANSPOSE REVERSES THE ORDER OF THE DIMENSIONS.
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND COLUMNS, THE PAGES WILL BECOME ROWS AND THE ROWS PAGES.
THAT IS, THE ELEMENT IN THE FIRST PAGE, SECOND ROW, THIRD COLUMN WILL BE TRANSPOSED TO THE THIRD PAGE, SECOND ROW, FIRST COLUMN.

```
IF ARRAY[1;2;3;4] => 7   THEN   (⍉ARRAY)[4;3;2;1]  => 7
```

THE SHAPE OF THE RESULT WILL ALWAYS BE THE REVERSE OF THE SHAPE OF THE ARGUMENT. IF ⍴ARRAY => 3 2 7 THEN ⍴⍉ARRAY => 7 2 3

..... ⍉ .....

THE SYMBOL ⍉ (ALT 6) IS USED FOR THE DYADIC TRANSPOSE FUNCTION.
THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK.
THE LEFT ARGUMENT MUST BE A POSITIVE INTEGER SCALAR OR VECTOR.
IT RETURNS ALL THE ELEMENTS OF THE RIGHT ARGUMENT IN A DIFFERENT SHAPE AND ORDER, AS FOLLOWS.

USING ⍉ WITH A SCALAR RIGHT ARGUMENT RETURNS A 'LENGTH ERROR' UNLESS THE LEFT ARGUMENT IS AN EMPTY VECTOR. THEN IT RETURNS THE SCALAR.

1⍉V RETURNS THE VECTOR V. ANY OTHER SCALAR LEFT ARGUMENT USED WITH A VECTOR RIGHT ARGUMENT PRODUCES A 'DOMAIN ERROR'. ANY OTHER RANK LEFT ARGUMENT RESULTS IN A 'LENGTH ERROR'.

WITH A MATRIX RIGHT ARGUMENT, THE LEFT ARGUMENT MUST BE A VECTOR; EITHER (1 2) OR (2 1). (2 1)⍉M PRODUCES THE TRANSPOSE OF M; THE COLUMNS (THE SECOND DIMENSION) BECOME THE ROW (THE FIRST DIMENSION). (1 2)⍉ M ⇒ M

WITH A RIGHT ARGUMENT OF RANK R, THE LEFT ARGUMENT MUST CONTAIN ALL THE INTEGERS FROM 1 TO R, IN ANY ORDER.
TRANSPOSE WILL TAKE THE DIMENSIONS OF THE RIGHT ARGUMENT AND PUT THEM IN THE ORDER SPECIFIED BY THE LEFT ARGUMENT.
FOR EXAMPLE, IF ⍴A ⇒ 2 3 4 THEN (3 1 2)⍉A WILL PUT THE THIRD DIMENSION FIRST (COLUMNS BECOME PAGES), THE FIRST DIMENSION SECOND (PAGES BECOME ROWS), AND THE SECOND DIMENSION THIRD (ROWS BECOME COLUMNS). SO ⍴(3 1 2)⍉A ⇒ 4 2 3.
IN OTHER WORDS, THE ELEMENT IN THE SECOND COLUMN, THIRD ROW, FIRST PAGE WILL BE TRANSPOSED TO THE SECOND PAGE, THIRD COLUMN, FIRST ROW, AND SO ON. IF A[1 3 2] ⇒ 8 THEN ((3 1 2)⍉A)[2 1 3] ⇒ 8.

..... ↑ .....

THE SYMBOL ↑ (UPSHIFT Y) IS USED FOR THE DYADIC TAKE FUNCTION.

THE LEFT ARGUMENT MUST BE AN INTEGER SCALAR OR VECTOR.

THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK.

TAKE RETURNS SELECTED ELEMENTS OF THE RIGHT ARGUMENT AS FOLLOWS:

WHEN THE RIGHT ARGUMENT IS A SCALAR, THE LEFT ARGUMENT CAN BE A SCALAR OR A VECTOR OF ANY LENGTH.

THE RESULT OF L↑R HAS A SHAPE EXACTLY EQUAL TO L. R WILL BE THE FIRST ELEMENT IF L IS POSITIVE, AND THE LAST ELEMENT IF L IS NEGATIVE. THE OTHER ELEMENTS WILL BE 0 IF R IS NUMERIC, BLANK IF R IS CHARACTER.

FOR EXAMPLE: 1↑4 => 4 ⍴1↑4 => 1 2↑'A' => A

⍴2↑'A' => 2 ¯4↑2 => 0 0 0 2 ⍴(3 4)↑1 => 3 4

IF THE RIGHT ARGUMENT IS A VECTOR, THE LEFT ARGUMENT MUST BE A SCALAR OR A 'RANK ERROR' WILL RESULT.

S↑V WILL RETURN THE FIRST S ELEMENTS OF V, IF S IS POSITIVE, OR THE LAST S ELEMENTS OF V, IF S IS NEGATIVE.

IF S EXCEEDS THE LENGTH OF V, THE RESULT WILL BE PADDED WITH 0'S, OR BLANKS, ON THE RIGHT IF S IS POSITIVE, ON THE LEFT IF S IS NEGATIVE.

FOR EXAMPLE: 3↑⍳10 => 1 2 3 ¯3↑'ALPHABET' => BET

5↑'ABC' => ABC ⍴5↑'ABC' => 5 ¯4↑ 2 3 => 0 0 2 3

WHEN THE RIGHT ARGUMENT IS A MATRIX OR HIGHER LEVEL ARRAY, THE LEFT ARGUMENT MUST BE A VECTOR WITH AS MANY ELEMENTS AS THERE ARE DIMENSIONS IN THE RIGHT ARGUMENT, OR A 'LENGTH ERROR' RESULTS.

FOR EXAMPLE: IF M => 9 8 7 ¯1 2↑M => 6 5 2 4↑M => 9 8 7 0

6 5 4 6 5 4 0

THE FIRST ELEMENT OF THE VECTOR DETERMINES WHICH ROWS ARE TAKEN, AND THE SECOND ELEMENT DETERMINES WHICH COLUMNS ARE TAKEN.

THE RESULT WILL BE PADDED WITH 0'S OR BLANKS AS ABOVE.

FOR HIGHER LEVEL ARRAYS, EACH ELEMENT OF THE LEFT ARGUMENT WILL
TAKE ELEMENTS FROM THE CORRESPONDING DIMENSION.

FOR EXAMPLE, IF ⍴A ⇒ 6 5 4; (2 ¯3 1)↑M WILL RETURN THE FIRST
COLUMN OF THE LAST THREE ROWS OF THE FIRST TWO PAGES.

THE RESULT OF TAKE WILL ALWAYS HAVE A SHAPE EXACTLY EQUAL TO THE
ABSOLUTE VALUE OF THE LEFT ARGUMENT.

IF THERE IS A ZERO (0) ANYWHERE IN THE LEFT ARGUMENT, THE ANSWER WILL
BE AN EMPTY ARRAY WITH THE APPROPRIATE SHAPE.

FOR EXAMPLE; ⍴2 ¯3 1↑A ⇒2 3 1 ⍴7 0 7↑A ⇒ 7 0 7

..... ↓ .....

THE SYMBOL ↓ (UPSHIFT U) IS USED FOR THE DYADIC DROP FUNCTION.

THE LEFT ARGUMENT MUST BE AN INTEGER SCALAR OR VECTOR.

THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK.

DROP RETURNS SELECTED ELEMENTS OF THE RIGHT ARGUMENT AS FOLLOWS:

WHEN THE RIGHT ARGUMENT IS A SCALAR, THE LEFT ARGUMENT CAN BE A SCALAR OR A VECTOR OF ANY LENGTH.

IF ALL ELEMENTS OF THE LEFT ARGUMENT EQUAL 0, THE RESULT OF L↓S WILL BE AN ARRAY CONTAINING THE RIGHT ARGUMENT; OTHERWISE IT WILL BE AN EMPTY ARRAY. THIS ARRAY HAS AS MANY DIMENSIONS AS THE LEFT ARGUMENT HAS ELEMENTS AND EACH DIMENSION IS OF LENGTH 1.

FOR EXAMPLE: 1↓4 ⇒ (EMPTY) ⍴1↓4 ⇒ 1 0↓'A' ⇒ A

2 4 6↓2 ⇒ (EMPTY) ⍴(2 4 6)↓2 ⇒ 1 1 1 0 0 0↓3 ⇒ 3

IF THE RIGHT ARGUMENT IS A VECTOR, THE LEFT ARGUMENT MUST BE A SCALAR OR A 'RANK ERROR' WILL RESULT.

S↓V WILL RETURN V OMITTING THE FIRST S ELEMENTS, IF S IS POSITIVE, OR THE LAST S ELEMENTS, IF S IS NEGATIVE.

IF S EXCEEDS THE LENGTH OF V, THE RESULT WILL BE AN EMPTY VECTOR.

FOR EXAMPLE: 6↓⍳10 ⇒ 7 8 9 10 ¯3↓'ALPHABET' ⇒ ALPHA

5↓'ABC' ⇒ (EMPTY) ⍴5↓'ABC' ⇒ 0 0↓1 3 ⇒ 1 3

WHEN THE RIGHT ARGUMENT IS A MATRIX OR HIGHER LEVEL ARRAY, THE LEFT ARGUMENT MUST BE A VECTOR WITH AS MANY ELEMENTS AS THERE ARE DIMENSIONS IN THE RIGHT ARGUMENT, OR A 'LENGTH ERROR' RESULTS.

FOR EXAMPLE: IF M ⇒ 9 8 7 1 ¯1↓M ⇒ 6 5 0 2↓M ⇒ 7

6 5 4 4

THE FIRST ELEMENT OF THE VECTOR DETERMINES WHICH ROWS ARE DROPPED, AND THE SECOND ELEMENT DETERMINES WHICH COLUMNS ARE DROPPED.

FOR HIGHER LEVEL ARRAYS, EACH ELEMENT OF THE LEFT ARGUMENT WILL

DROP ELEMENTS FROM THE CORRESPONDING DIMENSION.

FOR EXAMPLE, IF ⍴A ⇒ 6 5 4; (2 ¯3 1)↓A WILL RETURN THE LAST THREE COLUMNS OF THE FIRST TWO ROWS OF THE LAST FOUR PAGES.

THE RESULT OF DROP WILL ALWAYS HAVE A SHAPE EXACTLY EQUAL TO THE SHAPE OF THE RIGHT ARGUMENT MINUS THE ABSOLUTE VALUE OF THE LEFT ARGUMENT, EXCEPT AS FOLLOWS;

IF ANY ELEMENT OF THE LEFT ARGUMENT EXCEEDS THE LENGTH OF THE CORRESPONDING DIMENSION IN THE RIGHT ARGUMENT, THE RESULT WILL BE AN EMPTY ARRAY OF THE APPROPRIATE SHAPE.

IN THE ABOVE EXAMPLE; ⍴(2 ¯3 1)↓A ⇒ 4 2 3

⍴(3 4 5)↓A ⇒ 3 1 0 3 4 5↓A ⇒ (EMPTY)

..... / .....

THE SYMBOL / IS USED FOR THE DYADIC COMPRESS FUNCTION,
THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK,
THE LEFT ARGUMENT MUST BE 1,0, OR A VECTOR OF 1'S AND 0'S,
COMPRESS RETURNS SELECTED ELEMENTS OF THE RIGHT ARGUMENT, AS FOLLOWS;

1/A WILL RETURN THE RIGHT ARGUMENT, WHILE 0/A RETURNS AN EMPTY ARRAY,

V1/V2 WILL RETURN THE ELEMENTS OF V2 THAT CORRESPOND TO 1'S IN V1,
FOR EXAMPLE: 1 0 1/1 2 3 => 1 3 1 1 0 1/'HALT' => HAT
TWO VECTOR ARGUMENTS MUST HAVE THE SAME NUMBER OF ELEMENTS OR A
'LENGTH ERROR' WILL RESULT,

V/M TAKES THE COLUMNS AS UNITS AND RETURNS THE COLUMNS IN M WHICH
CORRESPOND TO THE 1'S IN V, FOR EXAMPLE;

```
IF M => 1 2 3    THEN  1 0 1/M => 1 3
        4 5 6                     4 6
        7 8 9                     7 9
```

APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE COLUMNS,
SINCE COLUMNS ARE THE SECOND OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL / WORKS 'OVER' THE LAST DIMENSION,
TO COMPRESS OVER THE FIRST DIMENSION (E,G, ROWS OF A MATRIX), USE THE
SYMBOL ⌿ (ALT /), IN PLACE OF /,

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS,
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND
COLUMNS, / WILL COMPRESS OVER THE COLUMNS, ⌿ OVER THE PAGES,
TO COMPRESS OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN
BRACKETS FOLLOWING THE / (OR THE ⌿) SYMBOL,
THE LEFT ARGUMENT MUST BE A SCALAR, OR A VECTOR WITH THE SAME
NUMBER OF ELEMENTS AS THE DIMENSION BEING COMPRESSED OVER,
FOR EXAMPLE, TO COMPRESS OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY

CALLED AA WHERE ⍴AA ⍹ 2 3 4, ENTER S/[2]AA OR V/[2]AA OR S⌿[2]AA OR V⌿[2]AA WHERE ⍴V ⍹ 3. TO COMPRESS OVER THE PAGES, ENTER S⌿AA OR V⌿AA OR S/[1]AA OR V/[1]AA WHERE ⍴V ⍹ 2. TO COMPRESS OVER THE COLUMNS, ENTER S/AA OR V/AA OR S⌿[3]AA OR V⌿[3] AA WHERE ⍴V ⍹ 4.

..... ≠ .....

THE SYMBOL ≠ (ALT /) IS USED FOR THE DYADIC COMPRESS FUNCTION.

THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK.

THE LEFT ARGUMENT MUST BE 1,0, OR A VECTOR OF 1'S AND 0'S.

COMPRESS RETURNS SELECTED ELEMENTS OF THE RIGHT ARGUMENT, AS FOLLOWS:

1≠A WILL RETURN THE RIGHT ARGUMENT, WHILE 0≠A RETURNS AN EMPTY ARRAY.

V1≠V2 WILL RETURN THE ELEMENTS OF V2 THAT CORRESPOND TO 1'S IN V1.

FOR EXAMPLE: 1 0 1≠1 2 3 => 1 3 1 1 0 1≠'HALT' => HAT

TWO VECTOR ARGUMENTS MUST HAVE THE SAME NUMBER OF ELEMENTS OR A

'LENGTH ERROR' WILL RESULT.

V≠M TAKES THE ROWS AS UNITS AND RETURNS THE ROWS IN M WHICH

CORRESPOND TO THE 1'S IN V. FOR EXAMPLE:

IF M => 1 2 3 THEN 1 0 1≠M => 1 2 3

4 5 6 7 8 9

7 8 9

APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE ROWS.

SINCE ROWS ARE THE FIRST OF TWO DIMENSIONS OF A MATRIX,

THIS MEANS THAT THE SYMBOL / WORKS 'OVER' THE FIRST DIMENSION.

TO COMPRESS OVER THE LAST DIMENSION (E.G. COLUMNS OF A MATRIX),

USE THE SYMBOL / IN PLACE OF ≠.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS.

FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND

COLUMNS, / WILL COMPRESS OVER THE COLUMNS, ≠ OVER THE PAGES.

TO COMPRESS OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN

BRACKETS FOLLOWING THE ≠ (OR THE /) SYMBOL.

THE LEFT ARGUMENT MUST BE A SCALAR, OR A VECTOR WITH THE SAME

NUMBER OF ELEMENTS AS THE DIMENSION BEING COMPRESSED OVER.

FOR EXAMPLE, TO COMPRESS OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY

CALLED AA WHERE ⍴AA ⇒ 2 3 4, ENTER S⌿[2]AA OR V⌿[2]AA OR S/[2]AA OR V/[2]AA WHERE ⍴V ⇒ 3. TO COMPRESS OVER THE PAGES, ENTER S⌿AA OR V⌿AA OR S/[1]AA OR V/[1]AA WHERE ⍴V ⇒ 2. TO COMPRESS OVER THE COLUMNS, ENTER S/AA OR V/AA OR S⌿[3]AA OR V⌿[3] AA WHERE ⍴V ⇒ 4.

..... \ .....

THE SYMBOL \ (UPSHIFT /) IS USED FOR THE DYADIC EXPAND FUNCTION.
THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK.
THE LEFT ARGUMENT MUST BE A VECTOR OF 1'S AND 0'S.
IS NUMERIC, OR BLANKS IF IT IS CHARACTER, AS FOLLOWS:
EXPAND RETURNS THE RIGHT ARGUMENT, ADDING 0'S IF THE RIGHT ARGUMENT
IS NUMERIC, OR BLANKS IF IT IS CHARACTER, AS FOLLOWS:

V\S WILL RETURN A VECTOR CONSISTING OF THE SCALAR REPEATED AS MANY
TIMES AS THERE ARE 1'S IN V.
FOR EXAMPLE: 1 0 1\3 => 3 3
IF THERE ARE NO 1'S IN V, V\S RETURNS AN EMPTY VECTOR.

V1/V2 WILL RETURN THE ELEMENTS OF V2, WITH 0'S OR BLANKS INSERTED
TO CORRESPOND TO THE 0'S IN V1. FOR EXAMPLE:
1 1 0 1\1 2 3 => 1 2 0 3 1 0 1 0 1\'APL' => A P L
THE LEFT ARGUMENT MUST CONTAIN AS MANY 1'S AS THERE ARE ELEMENTS
IN THE RIGHT ARGUMENT OR A 'LENGTH ERROR' WILL RESULT.

V\M TAKES THE COLUMNS AS UNITS AND RETURNS THE A MATRIX WITH COLUMNS
OF 0'S OR BLANKS INSERTED TO CORRESPOND TO THE 1'S IN V.

```
IF M => 1 2 3    THEN   1 0 1 0 1\M => 1 0 2 0 3
        4 5 6                          4 0 5 0 6
        7 8 9                          7 0 8 0 9
```

APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE COLUMNS.
SINCE COLUMNS ARE THE SECOND OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL \ WORKS 'OVER' THE LAST DIMENSION.
TO EXPAND OVER THE FIRST DIMENSION (E.G. ROWS OF A MATRIX), USE THE
SYMBOL ⍀ (ALT ,), IN PLACE OF \.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS.
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND

COLUMNS, \ WILL EXPAND OVER THE COLUMNS, ⌿ OVER THE PAGES.
TO EXPAND OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN
BRACKETS FOLLOWING THE \ (OR THE ⌿) SYMBOL.
FOR EXAMPLE, TO EXPAND OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY
CALLED AA WHERE ⍴AA ⇒ 2 3 4, ENTER V\[2]AA OR V⌿[2]AA
WHERE +/V ⇒ 3. TO EXPAND OVER THE PAGES, ENTER V⌿AA OR V\[1]AA
WHERE +/V ⇒ 2. TO EXPAND OVER THE COLUMNS, ENTER V\AA OR
V⌿[3]AA WHERE +/V ⇒ 4

..... ⍀ .....

THE SYMBOL ⍀ (ALT ,) IS USED FOR THE DYADIC EXPAND FUNCTION.
THE RIGHT ARGUMENT CAN BE CHARACTER OR NUMERIC, OF ANY RANK.
THE LEFT ARGUMENT MUST BE A VECTOR OF 1'S AND 0'S.
EXPAND RETURNS THE RIGHT ARGUMENT, ADDING 0'S IF THE RIGHT ARGUMENT
IS NUMERIC, OR BLANKS IF IT IS CHARACTER, AS FOLLOWS:

V⍀S WILL RETURN A VECTOR CONSISTING OF THE SCALAR REPEATED AS MANY
TIMES AS THERE ARE 1'S IN V.
FOR EXAMPLE: 1 0 1⍀3 ⇒ 3 3
IF THERE ARE NO 1'S IN V, V⍀S RETURNS AN EMPTY VECTOR.

V1⍀V2 WILL RETURN THE ELEMENTS OF V2, WITH 0'S OR BLANKS INSERTED
TO CORRESPOND TO THE 0'S IN V1. FOR EXAMPLE:
1 1 0 1⍀1 2 3 ⇒ 1 2 0 3 1 0 1 0 1⍀'APL' ⇒ A P L
THE LEFT ARGUMENT MUST CONTAIN AS MANY 1'S AS THERE ARE ELEMENTS
IN THE RIGHT ARGUMENT OR A 'LENGTH ERROR' WILL RESULT.

V⍀M TAKES THE ROWS AS UNITS AND RETURNS THE MATRIX WITH ROWS
OF 0'S OR BLANKS INSERTED TO CORRESPOND TO THE 1'S IN V.
IF M ⇒ 1 2 3 THEN 1 0 1 ⍀ M ⇒ 1 2 3
4 5 6 0 0 0
4 5 6

APL USERS REFER TO THIS AS WORKING 'ACROSS' OR 'OVER' THE ROWS.
SINCE ROWS ARE THE FIRST OF TWO DIMENSIONS OF A MATRIX,
THIS MEANS THAT THE SYMBOL ⍀ WORKS 'OVER' THE FIRST DIMENSION.
TO EXPAND OVER THE LAST DIMENSION (E.G. COLUMNS OF A MATRIX), USE
THE SYMBOL \ (UPSHIFT /), IN PLACE OF ⍀.

THIS IS IMPORTANT WHEN WORKING WITH HIGHER-DIMENSIONAL ARRAYS.
FOR EXAMPLE, IN A THREE-DIMENSIONAL ARRAY OF PAGES, ROWS, AND
COLUMNS, \ WILL EXPAND OVER THE COLUMNS, ⍀ OVER THE PAGES.

TO EXPAND OVER ANY DIMENSION, YOU MAY SPECIFY THE DIMENSION IN BRACKETS FOLLOWING THE ⌿ (OR THE \) SYMBOL.

THE LEFT ARGUMENT MUST BE A VECTOR WITH THE SAME NUMBER OF 1'S AS THERE ARE ELEMENTS IN THE DIMENSION BEING EXPANDED OVER.

FOR EXAMPLE, TO EXPAND OVER THE ROWS OF A THREE-DIMENSIONAL ARRAY CALLED AA WHERE ⍴AA ⇒ 2 3 4, ENTER V⌿[2]AA OR V\[2]AA WHERE +/V ⇒ 3. TO EXPAND OVER THE PAGES, ENTER V⌿AA OR V\[1]AA WHERE +/V ⇒ 2. TO EXPAND OVER THE COLUMNS, ENTER V\AA OR V⌿[3]AA WHERE +/V ⇒ 4

..... , ...

THE SYMBOL , FOLLOWED BY AN NUMBER IN BRACKETS IS USED FOR THE DYADIC LAMINATE FUNCTION, AN EXTENSION OF THE CATENATE FUNCTION. THIS FUNCTION WILL COMBINE TWO CHARACTER OR NUMERIC ARGUMENTS OF ANY RANK (EXCEPT TWO SCALARS) INTO A SINGLE ARRAY.

VECTORS CAN BE EXTENDED BY LAMINATING S,[1]V OR V,[1]S OR V,[1]V. THE NUMBER IN THE BRACKETS MUST BE 1 BECAUSE VECTOR HAVE ONLY ONE DIMENSION. THIS WORKS EXACTLY LIKE CATENATE.

NOW FOR SOMETHING REALLY TRICKY:
TO MAKE A TWO-ROW MATRIX OUT OF TWO VECTORS OF THE SAME LENGTH, ENTER V1,[.5]V2. ANY INDEX BETWEEN 0 AND 1 WILL DO. THIS CREATES A NEW DIMENSION PRECEDING THE FIRST DIMENSION OF THE ARGUMENTS. THE OTHER DIMENSION STAYS THE SAME LENGTH.
TO MAKE A TWO-COLUMN MATRIX, ENTER V1,[1.5]V2 (ANY INDEX FROM 1 TO 2). NOW THE NEW DIMENSION COMES AFTER THE FIRST, WHICH IS THE SAME AS THE LENGTH OF THE ARGUMENTS.

LAMINATING A SCALAR TO A MATRIX WILL RESULT IN A MATRIX WITH ONE MORE ROW (IF THE INDEX NUMBER IN BRACKETS IS 1) OR ONE MORE COLUMN (IF THE INDEX NUMBER IS 2) THAN THE ORIGINAL MATRIX. ALL ENTRIES IN THIS ROW OR COLUMN WILL HAVE THE VALUE OF THE SCALAR.

```
      IF M => 1 2  THEN   5,[1]M  =>  5 5  AND  M,[2]5  =>  1 2 5
              3 4                     1 2                   3 4 5
                                      3 4
```

LAMINATING TWO MATRIXES WILL RESULT IN A MATRIX CONTAINING THE TWO ORIGINAL MATRIXES SIDE BY SIDE (INDEX 2) OR VERTICALLY (INDEX 1).

```
FOR EXAMPLE:   IF M1 => 1 2  AND  M2 => 5 6
                        3 4             7 8
   THEN  M1,[2]M2  =>  1 2 5 6  AND  M2,[1]M1  =>  1 2
```

```
      3 4 7 8                3 4
                             5 6
                             7 8
```

TO STACK THE MATRIXES BY CREATING A THIRD DIMENSION, ENTER
M1,[.5]M2 (ANY INDEX BETWEEN 0 AND 1). YOU CAN ALSO CREATE
A THIRD DIMENSION BY M1,[1.5]M2 OR M1,[2.5]M2. TRY IT.

VARIATIONS ON THIS THEME ARE ALMOST ENDLESS--EXPERIMENT!
IN EVERY CASE, THE NUMBER IN THE BRACKETS REFERS TO THE DIMENSION
WHERE THE 'SEAM' IS--THE DIMENSION THAT INCREASES IN LENGTH.
IF THIS IS A NEW DIMENSION, IT WILL ALWAYS BE LENGTH 2 IN THE RESULT.
THE OTHER DIMENSIONS MUST BE THE SAME LENGTH IN BOTH ARGUMENTS
(UNLESS ONE IS A SCALAR) OR A 'LENGTH ERROR' WILL RESULT.

..... ⌹ .....

THE SYMBOL ⌹ (ALT X) IS USED FOR THE MONADIC MATRIX INVERSE FUNCTION.
IT TAKES A NUMERIC SCALAR, VECTOR, OR MATRIX ARGUMENT, AND RETURNS
THE LEFT INVERSE. IN TERMS OF MATRIX ALGEBRA, THIS IS THE MATRIX
WHICH PRODUCES THE IDENTITY WHEN LEFT-MULTIPLIED WITH THE ORIGINAL.

WHEN USED WITH SCALARS, ⌹ IS EQUIVALENT TO ÷.

⌹ WILL TREAT VECTORS AS 1×L MATRIXES. ⌹V WILL RETURN THE VECTOR VI
SO THAT VI +.× V ⇒ 1

SQUARE MATRIXES WILL BE INVERTED NORMALLY.
FOR EXAMPLE: IF M ⇒ 1 2 ⌹M ⇒ ¯2 1
3 4 1.5 ¯0.5

SOME NON-SQUARE MATRIXES MAY ALSO BE INVERTED.
FOR EXAMPLE: IF M2 ⇒ 1 4 THEN ⌹M2 ⇒ ¯0.9444 ¯0.1111 0.7222
2 5 0.4444 0.1111 ¯0.2222
3 6
IF M IS NOT INVERTIBLE, A 'DOMAIN ERROR' WILL RESULT.

..... ⌹ .....

THE SYMBOL ⌹ (ALT x) IS USED FOR THE DYADIC MATRIX DIVIDE FUNCTION.

IT TAKES ARITHMETIC ARGUMENTS WITH RANK ≤ 3. THE TWO ARGUMENTS
MUST HAVE THE SAME NUMBER OF ROWS.

IN TERMS OF MATRIX ALGEBRA, IT LEFT MULTIPLIES THE LEFT ARGUMENT BY
THE LEFT INVERSE OF THE RIGHT ARGUMENT.

B⌹A PRODUCES THE SOLUTION TO THE SET OF LINEAR EQUATIONS AX←B.

WHEN USED WITH TWO SCALARS, ⌹ IS EQUIVALENT TO ÷.

WHEN USED WITH VECTORS, ⌹ OPERATES AS IF THE VECTORS ARE 1×C OR R×1
MATRIXES, WHICHEVER IS APPROPRIATE.

..... °. .....

THE SYMBOLS ∘. (UPSHIFT J FOLLOWED BY A PERIOD) ARE PLACED PRECEDING
ANY DYADIC FUNCTION SYMBOL TO PRODUCE THE OUTER PRODUCT FUNCTION.
THE ARGUMENTS MAY BE NUMERIC OR CHARACTER, IF THE DYADIC SYMBOL
ACCEPTS CHARACTER ARGUMENTS, THEY MAY BE OF ANY RANK AND DO NOT
HAVE TO AGREE IN SHAPE.
THE OUTER PRODUCT RESULTS IN THE DYADIC SYMBOL OPERATING ON EACH
ELEMENT OF THE LEFT ARGUMENT COMBINED WITH EVERY ELEMENT OF
THE RIGHT ARGUMENT.

```
FOR EXAMPLE:  1 2 ∘.× 3 4 5   ⇒   3  4  5
                                  6  8 10

              'STOP' ∘.= 'POST'  ⇒   0 0 1 0
                                     0 0 0 1
                                     0 1 0 0
                                     1 0 0 0
```

OUTER PRODUCT (ALSO CALLED 'JOT DOT' BECAUSE OF THE SYMBOLS)
CAN BE EXTENDED TO ARRAYS OF ANY DIMENSION OR SIZE.
THE SHAPE OF THE RESULT WILL ALWAYS BE EQUAL TO THE SHAPE OF THE
LEFT ARGUMENT CATENATED TO THE SHAPE OF THE RIGHT ARGUMENT.
FOR EXAMPLE: IF ⍴A1 ⇒ 3 4 AND ⍴A2 ⇒ 6 5
THEN ⍴A1∘.+A2 ⇒ 3 4 6 5

```
.............................  .  ................................
```

THE SYMBOL . (PERIOD OR 'DOT') MAY BE PLACED BETWEEN ANY TWO DYADIC FUNCTION SYMBOLS TO CREATE THE INNER PRODUCT FUNCTION.

THE ARGUMENTS MAY BE NUMERIC OR CHARACTER, IF APPROPRIATE, OF ANY RANK. THE LENGTH OF THE LAST DIMENSION OF THE LEFT ARGUMENT MUST BE THE SAME AS THE LENGTH OF THE FIRST DIMENSION OF THE RIGHT ARGUMENT.

INNER PRODUCT IS MOST COMMONLY USED WITH + AND × TO FORM THE CONVENTIONAL DOT PRODUCT OR MATRIX PRODUCT OF MATRIX ALGEBRA.

FOR EXAMPLE: IF V1 ⇒ 1 2 3 AND V2 ⇒ 4 5 6
THEN V1 +.× V2 ⇒ 32

INNER PRODUCT MAY BE EXTENDED TO HIGHER LEVEL ARRAYS AS WELL AS MATRIXES. THE SHAPE OF THE RESULT WILL BE THE SAME AS THE SHAPE OF THE LEFT ARGUMENT CATENATED TO THE SHAPE OF THE RIGHT ARGUMENT, OMITTING THE MATCHING DIMENSIONS (LAST ON THE LEFT, FIRST ON THE RIGHT.

FOR EXAMPLE: IF ⍴A1 ⇒ 3 6 4 AND ⍴A2 ⇒ 4 5
THEN ⍴ A1 +.× A2 ⇒ 3 6 5

..... ⍳ .....

THE SYMBOL ⍳ (UPSHIFT I) IS USED FOR THE DYADIC INDEX OF FUNCTION.
THE LEFT ARGUMENT MUST BE A VECTOR AND THE RIGHT ARGUMENT CAN BE
ANY RANK. INDEX OF CAN BE USED WITH CHARACTER OR NUMERIC DATA.
IT RETURNS THE POSITION IN THE LEFT ARGUMENT OF THE ELEMENTS OF THE
RIGHT ARGUMENT. THE SHAPE OF THE RESULT IS THE SAME AS THE
SHAPE OF THE RIGHT ARGUMENT.
IF THE RIGHT ARGUMENT DOES NOT OCCUR IN THE LEFT ARGUMENT, THE RESULT
WILL BE ONE PLUS THE LENGTH OF THE LEFT ARGUMENT.

```
FOR EXAMPLE:  4 3 2 1⍳3    ⇒   2   BECAUSE 3 IS IN THE SECOND POSITION
              'HELLO'⍳'P'  ⇒   6   BECAUSE P IS NOT FOUND
```

V⍳A CHECKS EACH ELEMENT OF THE RIGHT ARGUMENT INDIVIDUALLY.

```
FOR EXAMPLE:  3 5 7 9⍳1 2 3 4 5  ⇒  5 5 1 5 2
   IF YOU ASSIGN   ALPHABET←'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
   THEN ALPHABET⍳'APL'  ⇒  1 16 12
```

..... ⍷ .....

THE SYMBOL ⍷ (UPSHIFT N) IS USED FOR THE DYADIC ENCODE FUNCTION.

IT TAKES NUMERIC ARGUMENTS OF ANY RANK AND RETURNS THE REPRESENTATION OF THE RIGHT ARGUMENT IN THE NUMBER SYSTEM SPECIFIED BY THE LEFT ARGUMENT.

THE ENTRIES IN THE LEFT ARGUMENT DEFINE THE NUMBER OF VALUES THAT CAN BE PLACED IN THAT POSITION.

FOR EXAMPLE, IN THE BINARY SYSTEM, EACH POSITION CAN TAKE ONE OF TWO VALUES, 0 OR 1. THEREFORE, TO TRANSLATE 133 INTO BINARY, ENTER:

```
2 2 2 2 2 2 2 2⍷133   ⇒   1 0 0 0 0 1 0 1
```

THE RESULT WILL ALWAYS BE THE SAME SHAPE AS THE LEFT ARGUMENT, SO BE SURE THAT THE LEFT ARGUMENT IS LARGE ENOUGH TO BE CORRECT.

FOR EXAMPLE: `2 2 2⍷133  ⇒  1 0 1`

ENCODE CAN ALSO BE USED TO TRANSLATE TIMES. GIVEN TIME IN SECONDS, ENTER `(2000 365 24 60 60) ⍷ TIME` TO RECEIVE A VECTOR OF YEARS (0 TO 1999), DAYS (0 TO 364), HOURS (0 TO 23), MINUTES (0 TO 60), AND SECONDS (0 TO 60).

FOR EXAMPLE: `2000 365 24 60 60⍷456701  ⇒  0 5 6 51 41`

..... ⊥ .....

THE SYMBOL ⊥ (UPSHIFT 9) IS USED FOR THE DYADIC DECODE FUNCTION,
IT TAKES NUMERIC ARGUMENTS OF ANY RANK AND RETURNS THE DECIMAL VALUE
OF THE RIGHT ARGUMENT WHICH IS WRITTEN IN THE NUMBER SYSTEM
SPECIFIED BY THE LEFT ARGUMENT,
THE ENTRIES IN THE LEFT ARGUMENT DEFINE THE VALUES OF EACH POSITION
IN THE NUMBER SYSTEM,

FOR EXAMPLE, IN THE BINARY SYSTEM, EACH POSITION CAN TAKE ONE OF TWO
VALUES, 0 OR 1, THEREFORE, TO TRANSLATE 10000101 FROM BINARY:

```
2 ⊥ 1 0 0 0 0 1 0 1   ⇒  133
```

UNLESS ONE ARGUMENT IS A SCALAR, THE ARGUMENTS MUST BE THE SAME SHAPE
OR A 'LENGTH ERROR' WILL BE PRODUCED,

DECODE CAN ALSO BE USED TO TRANSLATE TIMES, GIVEN TIME IN YEARS,
(0 TO 1999), DAYS (0 TO 364), HOURS (0 TO 23), MINUTES (0 TO 60),
AND SECONDS (0 TO 60), ENTER 2000 365 24 60 60 ⊥ TIME TO
FIND THE TIME IN SECONDS,

FOR EXAMPLE: 2000 365 24 60 60 ⊥ 0 5 6 51 41 ⇒ 456701

..... ⍎ .....

THE SYMBOL ⍎ (ALT [) IS USED FOR THE MONADIC EXECUTE FUNCTION,
IT TAKES A VECTOR CHARACTER ARGUMENT ONLY, AND TREATS IT AS
AN APL EXPRESSION,

FOR EXAMPLE: ⍎'1 + 2' ⇒ 3
IF V ⇒ 'HELP', ⍎V WILL RUN THE HELP FUNCTION
HOWEVER, ⍎ WILL NOT EXECUTE EXPRESSIONS BEGINNING WITH) OR ∇,

..... ⍕ .....

THE SYMBOL ⍕ (ALT]) IS USED FOR THE DYADIC FORMAT FUNCTION.
THE RIGHT ARGUMENT MUST BE NUMERIC, OF ANY RANK, AND THE LEFT
ARGUMENT MUST BE A VECTOR OF INTEGERS.
THIS FUNCTION CONVERTS THE RIGHT ARGUMENT TO CHARACTER DATA IN
A TABULAR FORMAT SPECIFIED BY THE LEFT ARGUMENT, AS FOLLOWS:

IF THE LEFT ARGUMENT IS A SCALAR, THE RIGHT ARGUMENT WILL BE WRITTEN
WITH THAT NUMBER OF DECIMAL PLACES.
FOR EXAMPLE: 2⍕14 ⇒ 14.00 (REMEMBER THIS IS NOW CHARACTER DATA)
3⍕15 28.35 0.1475 ⇒ 15.000 28.350 .148
NOTICE THAT THE COLUMNS ARE OF EQUAL WIDTH, SEPARATED BY ONE SPACE.

IF THE LEFT ARGUMENT IS A VECTOR OF LENGTH TWO, EACH COLUMN IN THE
RIGHT ARGUMENT WILL TAKE UP THE NUMBER OF SPACES SPECIFIED BY THE
FIRST ELEMENT OF THE LEFT ARGUMENT, AND EACH ENTRY WILL HAVE THE
NUMBER OF DECIMAL PLACES SPECIFIED BY THE SECOND ENTRY IN THE
LEFT ARGUMENT. FOR EXAMPLE:

```
IF  M ⇒ 1.2 3        THEN  5 1⍕M ⇒    1.2  3.1
       ¯4   5.55                      ¯4.0  5.5
```

THE COLUMN ELEMENT MUST BE LARGE ENOUGH TO WRITE THE LONGEST ENTRY,
INCLUDING ONE SPACE EACH FOR A NEGATIVE SIGN OR DECIMAL POINT.
IF IT IS NOT, A 'LENGTH ERROR' WILL RESULT.
REMEMBER TO ALLOW FOR A BLANK BETWEEN COLUMNS: 3 1⍕1 2 ⇒ 1.02.0

THE LEFT ARGUMENT MAY ALSO BE A VECTOR OF LENGTH EQUAL TO EXACTLY
TWICE THE NUMBER OF COLUMNS IN THE RIGHT ARGUMENT.
IN THIS CASE, THE ENTRIES IN THE LEFT ARGUMENT ARE PAIRED AND EACH
PAIR GOVERNS THE FORMAT OF THE CORRESPONDING COLUMN, AS ABOVE.
6 2 6 1 10 0 ⍕ 113.524 1 300.24 ⇒ 113.52 1.0 300

ONE FINAL TWIST: IF THE LEFT ARGUMENT IS A NEGATIVE SCALAR, THE

RESULT WILL BE WRITTEN IN EXPONENTIAL NOTATION, WITH THE NUMBER OF DECIMAL PLACES IN THE MANTISSA EQUAL TO ONE LESS THAN THE ABSOLUTE VALUE OF THE LEFT ARGUMENT.

```
FOR EXAMPLE: ¯1↑3 200   →     3E00  2E02
             ¯3↑4.6167 10.56 400  →     4.62E00  1.06E01  4.00E02
```

..... ⍕ .....

THE SYMBOL ⍕ (ALT]) IS USED FOR THE MONADIC FORMAT FUNCTION,
IT TAKES A NUMERIC OR CHARACTER ARGUMENT OF ANY RANK AND CONVERTS
IT TO CHARACTER DATA.

THIS IS USEFUL IN CATENATING NUMERIC VARIABLES TO CHARACTER DATA,
FOR EXAMPLE: IF S ⇒ 3 AND V ⇒ 'GO TO LINE '
V,S WILL PRODUCE A 'DOMAIN ERROR' BUT V,⍕S ⇒ GO TO LINE 3

D. MAKING MODIFICATIONS

The variable HOWMODS is included in the TUTOR workspace to guide the advanced student, programmer, or instructor who wishes to expand, modify, or customize the TUTOR functions and variables. Some suggestions for improvement are discussed in chapter 3. Any ideas for modification of TUTOR are welcomed--please pass them along to Professor R. R. Read or the author of this thesis, in care of the Operations Analysis curricular office.

```
      ,HOWMODS
THE FOLLOWING ARE SOME TIPS ON MODIFYING THE TUTOR WORKSPACE:

TO MODIFY TEXT OF DESCRIPTIVE VARIABLES:
   XEDIT THE VARIABLE (CHECK MAT FOR SHORT NAME)
   BE SURE TO INCLUDE THE APPROPRIATE FIRST COLUMN CONTROL CHARACTERS:
      , FOR GENERAL DESCRIPTION
      ∘ FOR SCALAR DESCRIPTION AND EXAMPLES
      v FOR VECTOR DESCRIPTION AND EXAMPLES
      n FOR MATRIX DESCRIPTION AND EXAMPLES
      * FOR HIGHER-LEVEL ARRAYS DESCRIPTION AND EXAMPLES

TO CHANGE THE ORDER OF LESSONS:
   XEDIT MAT TO ASSIGN NEW LESSON NUMBERS,
   THEN, TO REORDER MAT ACCORDING TO LESSON NUMBERS, ENTER:  ORDERMAT

TO WRITE NEW LESSONS:
   WRITE A NEW TEXT VARIABLE, USING CONTROL CHARACTERS AS ABOVE,
   XEDIT MAT TO ADD A NEW ROW,  BE SURE TO INCLUDE:
      COL 1-3:    LESSON NUMBER
      COL 28-31:  NAME OF THE TEXT VARIABLE (LIMIT 4 CHARACTERS)
      COL 35-51:  FULL NAME OF THE LESSON (LIMIT 23 CHARACTERS)
   THE OTHER COLUMNS MAY BE LEFT BLANK AS THEY PERTAIN TO FUNCTION
      SYMBOLS ONLY,

TO CHANGE OR WRITE NEW QUESTIONS (NOT DRILL):
   XEDIT CUES TO CHANGE OR ADD A QUESTION,  BE SURE TO ENCLOSE THE
      TEXT OF THE QUESTION, BUT NOT VARIABLES, IN QUOTE MARKS,
      A CARRIAGE RETURN MAY BE INCLUDED BY USING THE VARIABLE CR,
      ANSWER GROUPS MAY BE SEPARATE VARIABLES SUCH AS AR, RK,
   XEDIT SHOW TO INCLUDE NEW QUESTIONS IN ANY QUESTION VECTOR QV,
      BE SURE TO INCLUDE THE CORRESPONDING ANSWER IN THE VECTOR AV,
      .
```

APPENDIX C
SAMPLE RUN

This appendix contains a printout of a session with the APL tutor. Student response to questions is preceded by a dot or period. (This is a function of the terminal and not part of the input.) Other student input is generally indented eight spaces and preceded by several blank lines. At least one example of each of the various ways to use the TUTOR is included. Also the various types of student response to questions and drills are demonstrated.

```
      )LOAD TUTOR
SAVED 11:11:15 12/14/83

YOU MAY USE THE APL TUTOR IN THREE WAYS:
(1)  ENTER:   HELP
     TO SELECT THE SYMBOLS THAT YOU WANT INFORMATION ABOUT.
(2)  ENTER:   TEACH
     TO SELECT THE SYMBOLS THAT YOU WANT INFORMATION AND DRILL ON.
(3)  ENTER:   MENU
     TO SEE A LIST OF SYMBOLS AND TOPICS.

IF YOU HAVE NEVER USED THE APL TUTOR BEFORE, ENTER:   START

TO SEE THESE INSTRUCTIONS AGAIN AT ANY TIME, ENTER:   HOW
      .



      START

WELCOME TO THE APL TUTOR.

THE PURPOSE OF THIS WORKSPACE IS TO INTRODUCE YOU TO
   'A PROGRAMMING LANGUAGE' BY DESCRIBING THE FUNCTIONS OF THE
   MANY SPECIAL APL SYMBOLS, AND BY OUTLINING THE PROCEDURES FOR
   DESIGNING YOUR OWN FUNCTIONS.


THE APL TUTOR ASSUMES YOU HAVE HAD LINEAR ALGEBRA AND TRIGINOMETRY.
   CALCULUS IS NOT NECESSARY.
IF YOU HAVE NEVER HAD ANY COMPUTER PROGRAMMING BEFORE, DON'T WORRY.
   YOU CAN START USING APL RIGHT AWAY, AS A SUPER-SOPHISTICATED
   CALCULATOR WITH MANY BUILT-IN FUNCTIONS.
IF YOU HAVE STUDIED OTHER COMPUTER PROGRAMMING LANGUAGES, RELAX.
```

APL IS NOT LIKE ANY OF THE OTHER MAJOR HIGH-LEVEL LANGUAGES. YOU CAN FORGET ABOUT DATA TYPES, INPUT/OUTPUT FORMATTING, AND MANY OF THE OTHER TEDIOUS DETAILS OF FORTRAN, PASCAL, ETC. AFTER YOU ARE SUFFICIENTLY FAMILIAR WITH THE CALCULATOR MODE OF APL, YOU CAN LEARN TO DEFINE YOUR OWN FUNCTIONS WITH EASE.

HERE IS SOME BASIC INFORMATION YOU WILL NEED TO KNOW IN ORDER TO UNDERSTAND THE APL TUTOR LESSONS.

MONADIC AND DYADIC FUNCTIONS

YOU ARE ALREADY FAMILIAR WITH SYMBOLS WHICH REPRESENT ARITHMETIC FUNCTIONS, SUCH AS + OR ÷. APL USES THESE SYMBOLS AND MANY OTHERS TO REPRESENT A VARIETY OF FUNCTIONS.

MANY FUNCTIONS, SUCH AS +, REQUIRE TWO ARGUMENTS, THAT IS, TWO INPUT NUMBERS. IN APL, THESE ARE CALLED DYADIC FUNCTIONS, AND THE SYMBOL IS PLACED BETWEEN THE ARGUMENTS; FOR EXAMPLE, 3+4.

OTHER FUNCTIONS, SUCH AS LN (NATURAL LOG), REQUIRE ONLY ONE ARGUMENT. THESE MONADIC FUNCTION SYMBOLS ARE PLACED TO THE LEFT OF THE DATA WHICH THEY ARE TO OPERATE ON. FOR EXAMPLE, LN 3 IN APL IS ⍟3.

DATA TYPES

APL DISTINGUISHES ONLY TWO TYPES OF DATA: NUMERIC AND CHARACTER. VERY SIMPLY, CHARACTER DATA ARE ENCLOSED IN QUOTES (') WHEN ENTERED. '2' IS CHARACTER DATA; 2 IS NUMERIC DATA. SOME FUNCTIONS WILL OPERATE ON BOTH TYPES OF DATA, SOME ONLY ON NUMERIC.

ARRAYS

THE GREATEST STRENGTH OF APL LIES IN ITS ABILITY TO TAKE AN ENTIRE ARRAY OF NUMBERS AS A SINGLE ARGUMENT. THUS TWO MATRIXES CAN BE ADDED BY ENTERING SIMPLY A+B, WITH NO SUBSCRIPTS, LOOPS, ETC.

THIS MAKES IT IMPORTANT TO KNOW THE RANK (THE NUMBER OF DIMENSIONS) OF DATA IN USE. A SINGLE NUMBER IS NORMALLY A SCALAR (RANK 0). A SERIES OF NUMBERS IS A VECTOR, A ONE-DIMENSIONAL ARRAY (RANK 1).

NUMBERS CAN ALSO BE ARRANGED IN ROWS AND COLUMNS, TO MAKE A MATRIX (RANK 2). MATRIXES CAN BE 'STACKED' TO MAKE UP THE PAGES OF A THREE-DIMENSIONAL ARRAY, OFTEN CALLED A BOOK (RANK 3). IN FACT, THERE IS NO LIMIT TO THE NUMBER OF DIMENSIONS IN AN APL ARRAY.

THE LENGTH OF THE DIMENSIONS IS ALSO EFFECTIVELY UNLIMITED. LENGTH REFERS TO THE NUMBER OF ELEMENTS IN A DIMENSION, FOR EXAMPLE, THE NUMBER OF ROWS. LENGTH MAY EVEN BE 0.

IN THIS TUTORIAL, A SCALAR MAY BE REFERRED TO AS S, L, OR R. A VECTOR MAY BE CALLED V, V1, OR V2, WHILE A MATRIX IS M, M1, OR M2. AN ARRAY, WHICH MAY BE A VECTOR, A MATRIX, OR AN ARRAY OF ANY HIGHER DIMENSION, WILL BE LABELLED A, A1, OR A2.

EXAMPLES

WHEN EXAMPLES ARE GIVEN IN THE TEXT OF A FUNCTION DESCRIPTION, THE SYMBOL => IS USED BETWEEN THE EXAMPLE INPUT AND THE EXAMPLE OUTPUT.

FOR EXAMPLE: 2+5 => 7

THIS SYMBOL CAN BE READ AS 'PRODUCES' OR 'RETURNS'.

ERRORS

YOU SHOULD NOT RECEIVE ANY ERROR MESSAGES WHILE USING APL TUTOR. HOWEVER, YOU MAY SEE A PHRASE LIKE 'VALUE ERROR' (WITHOUT QUOTE MARKS) WHEN YOU ARE EXPECTING A NORMAL RESPONSE.

OR IF YOU ACCIDENTALLY HIT THE ENTER KEY WHEN THE COMPUTER IS NOT EXPECTING AN INPUT, THE FUNCTION MAY END ABRUPTLY, LIKE THIS:

```
MENU [6]
```

IF EITHER OF THESE THINGS HAPPEN, BE SURE TO ENTER: → (UPSHIFT ←) THEN RESTART WITH ANY COMMAND.

HALTING A LESSON

YOU MAY STOP ANY LESSON AT ANY POINT WHERE A QUESTION IS ASKED BY ENTERING: STOP

TO SEE THIS INFORMATION AGAIN AT ANY TIME, ENTER: BASICS

```
YOU MAY START A LESSON IN THREE WAYS:
(1)  ENTER:   LESSON NEXTLESSON
     TO START THE LESSON FOLLWING THE LAST ONE YOU COMPLETED.
(2)  ENTER:   LESSON NNN      (WHERE NNN IS A 3-DIGIT NUMBER)
     TO START LESSON NUMBER NNN.
     TO SEE A LIST OF LESSON NUMBERS, ENTER:   MENU
(3)  ENTER:   TEACH
     TO SELECT THE SYMBOL THAT YOU WANT A LESSON ON.

TO SEE THIS INFORMATION AGAIN AT ANY TIME, ENTER:  HOWTEACH


TO START YOUR FIRST LESSON IN APL, ENTER:  LESSON 101
      .




       LESSON 101
.............................. ¯ ..................................
THE SYMBOL ¯ (UPSHIFT 2) IS USED ONLY TO ENTER NEGATIVE NUMBERS.
IT CAN BE USED ONLY IN FRONT OF SCALARS (SINGLE NUMBERS),
   NOT VARIABLE NAMES, ARITHMETIC EXPRESSIONS, OR ARRAYS.
TO CHANGE ARITHMETIC SIGN USE THE SYMBOL - (UPSHIFT +).

HOW MANY ARGUMENTS DOES THE NEGATIVE_NUMBER FUNCTION TAKE?
.1
RIGHT

WHAT RANK CAN THE RIGHT ARGUMENT OF NEGATIVE_NUMBER BE?
ENTER S FOR SCALAR
      V FOR VECTOR
```

```
      X FOR MATRIX
      A FOR ANY RANK
.S
RIGHT




CAN NEGATIVE_NUMBER TAKE CHARACTER ARGUMENTS?
.N
RIGHT

THIS IS THE END OF YOUR LESSON.
FOR THE NEXT LESSON IN SEQUENCE, ENTER:   LESSON NEXTLESSON
TO START ANOTHER LESSON, ENTER:   LESSON NNN
      .




       LESSON NEXTLESSON
.....................................+.....................................
THE SYMBOL + IS USED FOR THE MONADIC CONJUGATE FUNCTION.
THIS IS AN IDENTITY FUNCTION WHICH RETURNS ITS ARGUMENT, AFTER
   EVALUATION IF NECESSARY.  IT TAKES NUMERIC ARGUMENTS OF ANY RANK.

HOW MANY ARGUMENTS DOES THE CONJUGATE FUNCTION TAKE?
.STOP

THIS LESSON HAS BEEN HALTED.
TO START ANOTHER LESSON, ENTER:   LESSON NNN
      .
```

```
      LESSON 109
.............................. ÷ ..................................
THE SYMBOL ÷ (UPSHIFT x) IS USED FOR THE DYADIC DIVISION FUNCTION.
IT PERFORMS SIMPLE DIVISION ON NUMERIC ARGUMENTS OF ANY RANK.

HOW MANY ARGUMENTS DOES THE DIVIDE FUNCTION TAKE?
.TWO
SORRY, PLEASE TRY AGAIN

HOW MANY ARGUMENTS DOES THE DIVIDE FUNCTION TAKE?
.2
RIGHT

WHAT RANK CAN THE LEFT ARGUMENT OF DIVIDE BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
.S
SORRY, PLEASE TRY AGAIN

WHAT RANK CAN THE LEFT ARGUMENT OF DIVIDE BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
.A
RIGHT

WHAT RANK CAN THE RIGHT ARGUMENT OF DIVIDE BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
```

```
,S
SORRY, PLEASE TRY AGAIN

WHAT RANK CAN THE RIGHT ARGUMENT OF DIVIDE BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
,V
SORRY, PLEASE TRY AGAIN

WHAT RANK CAN THE RIGHT ARGUMENT OF DIVIDE BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
,X
SORRY, THE CORRECT ANSWER IS A
YOU ENTERED 5 WRONG ANSWERS OUT OF 3 QUESTIONS.
IF YOU WANT TO RETAKE THIS LESSON, ENTER:   LESSON 109

THIS LESSON HAS BEEN HALTED.
TO START ANOTHER LESSON, ENTER:   LESSON NNN
      .
```

```
      HELP
ENTER THE SYMBOL(S) YOU WOULD LIKE INFORMATION ABOUT,
   OR...FOR MENU SELECTION, ENTER:  MENU
.MENU
INFORMATION IS AVAILABLE ON THE FOLLOWING SYMBOLS/TOPICS:
101 ¯ NEGATIVE_NUMBER   102 + CONJUGATE        103 + PLUS
104 - CHANGE_SIGN       105 - MINUS            106 × SIGNUM
107 × TIMES             108 ÷ RECIPROCAL       109 ÷ DIVIDE
121 ( PARENS            122 ' QUOTE            123 ← SPECIFICATION
124 ⍴ SHAPE             125 ⍴ RESHAPE          126 [ BRACKETS
127 , RAVEL             128 , CATENATE         129 ⍳ INDEX_GENERATOR
131 / REDUCTION         132 ⌿ REDUCTION        141 * EXPONENTIAL
142 * POWER             143 ⍟ NATURAL_LOG      144 ⍟ LOGARITHM
145 ○ PI_TIMES          146 ○ GEOMETRIC        147 ! FACTORIAL
148 ! BINOMIAL          149 | MAGNITUDE        151 \ SCAN
152 ⍀ SCAN              161 ⌈ CEILING          162 ⌊ FLOOR
163 ⌈ MAXIMUM           164 ⌊ MINIMUM          165 | RESIDUE
166 ⍋ GRADE_UP          167 ⍒ GRADE_DOWN       168 ? ROLL
169 ? DEAL              180 = EQUAL            181 ≠ NOT_EQUAL
182 < LESS_THAN         183 ≤ LESS_OR_EQUAL    184 ≥ GREATER_OR_EQUAL
185 > GREATER_THAN      191 ~ NOT              192 ∧ AND
193 ⍲ NAND              194 ∨ OR               195 ⍱ NOR
196 ∈ MEMBER OF         201 ⌽ REVERSE          202 ⊖ REVERSE
203 ⌽ ROTATE            204 ⊖ ROTATE           205 ⍉ TRANSPOSE-MONADIC
206 ⍉ TRANSPOSE-DYADIC  211 ↑ TAKE             212 ↓ DROP
213 / COMPRESS          214 ⌿ COMPRESS         215 \ EXPAND
216 ⍀ EXPAND            219 , LAMINATE         231 ⌹ MATRIX_INVERSE
232 ⌹ MATRIX_DIVIDE     241 ∘ OUTER_PRODUCT    242 . INNER_PRODUCT
301 ⍳ INDEX_OF          304 ⊤ ENCODE           305 ⊥ DECODE
306 ⍎ EXECUTE           307 ⍕ FORMAT-DYADIC    308 ⍕ FORMAT-MONADIC
TO SEE MENU AGAIN, ENTER:                MENU
FOR INFORMATION ABOUT FUNCTION, ENTER:   INFO NNN   (WHERE NNN IS MENU N
      UMBER)
TO GO THROUGH TUTORIAL LESSON, ENTER:    LESSON NNN
      .
```

```
      LESSON 107
.................................. x ....................................
THE SYMBOL x IS USED FOR THE DYADIC TIMES FUNCTION.
THIS PERFORMS SIMPLE MULTIPLICATION ON NUMERIC ARGUMENTS OF ANY RANK.

HOW MANY ARGUMENTS DOES THE TIMES FUNCTION TAKE?
.2
RIGHT

WHAT RANK CAN THE LEFT ARGUMENT OF TIMES BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
.
SORRY, THE CORRECT ANSWER IS A

WHAT RANK CAN THE RIGHT ARGUMENT OF TIMES BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
.ANY
RIGHT

FOR EXAMPLE: 2x4  =>  8       ORDER AND SPACING ARE NOT IMPORTANT.

DOES S1xS2  EQUAL  S2xS1?
.Y
RIGHT

WHAT IS THE RESULT OF:
 ¯20 x  80
.¯160
```

```
   INSTRUCTOR BEFORE RETURNING TO THIS UNIT.

THIS LESSON HAS BEEN HALTED.
TO START ANOTHER LESSON, ENTER:   LESSON NNN
      .
```

```
       LESSON 124
..... ⍴ .....
THE SYMBOL ⍴ IS USED FOR THE MONADIC SHAPE FUNCTION.
THIS RETURNS A DESCRIPTION OF THE SIZE OF ITS ARGUMENT, WHICH CAN
   BE NUMERIC OR CHARACTER, OF ANY RANK.
THE SYMBOL CAN BE USED TWICE (⍴⍴A) TO RETURN THE RANK OF ITS ARGUMENT.

HOW MANY ARGUMENTS DOES THE SHAPE FUNCTION TAKE?
.1
RIGHT

WHAT RANK CAN THE RIGHT ARGUMENT OF SHAPE BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
.A
RIGHT

IF THE ARGUMENT IS A SCALAR, ⍴S RETURNS NO NUMBER (AN EMPTY VECTOR),
   BECAUSE A SCALAR HAS NO DIMENSION.    ⍴⍴S  ⍝  0
```

ENTERING ⍴V RETURNS ONE NUMBER WHICH REPRESENTS THE LENGTH OF
THE VECTOR'S ONE DIMENSION, ⍴⍴V => 1

WHAT IS THE RESULT OF:
⍴ 10 20 90 ¯40
.⍴4
SORRY, THIS ANSWER NOT ACCEPTABLE, PLEASE ENTER A NUMERIC ANSWER.

WHAT IS THE RESULT OF:
⍴ 10 20 90 ¯40
.4
CORRECT!

WHAT IS THE RESULT OF:
⍴ 2 0 3
.3
CORRECT!

THE SHAPE OF MATRIXES IS EXPRESSED AS A TWO-ELEMENT VECTOR,
THE FIRST ELEMENT EQUALS THE NUMBER OF ROWS (THE LENGTH OF THE
FIRST DIMENSION), AND THE SECOND ELEMENT EQUALS THE NUMBER OF
COLUMNS (THE LENGTH OF THE SECOND DIMENSION).
FOR EXAMPLE: IF M => 1 2 3 THEN ⍴M => 2 3 AND ⍴⍴M => 2
4 5 6

FOR HIGHER-LEVEL ARRAYS, ⍴A WILL BE A VECTOR WITH AS MANY ELEMENTS AS
A HAS DIMENSIONS, THE NUMBER OF COLUMNS WILL ALWAYS BE THE LAST
ELEMENT OF ⍴A, THE NUMBER OF ROWS WILL ALWAYS BE THE NEXT-TO-LAST
ELEMENT, PRECEDED BY THE NUMBER OF 'PAGES' OR PLANES, PRECEDED BY
THE NUMBER OF 'BOOKS' OR SPACES, AND SO ON.
FOR EXAMPLE: IF D IS A 4-DIMENSIONAL ARRAY, ⍴⍴D => 4.
IF ⍴D => 5 3 4 2 THEN D HAS 5 BOOKS, EACH WITH 3 PAGES, EACH OF
WHICH HAS 4 ROWS AND 2 COLUMNS. (THIS WOULD BE DISPLAYED AS
FIFTEEN SUCCESSIVE 4x2 MATRIXES--THE FIRST THREE BEING THE FIRST

```
   BOOK, THE SECOND THREE BEING THE SECOND BOOK, ETC,)

CAN SHAPE TAKE CHARACTER ARGUMENTS?
.Y
RIGHT

IF SO, WHICH ARGUMENT(S) CAN BE CHARACTER?
ENTER R FOR RIGHT,
      L FOR LEFT, OR
      B FOR BOTH
.R
RIGHT

THIS IS THE END OF YOUR LESSON,
FOR THE NEXT LESSON IN SEQUENCE, ENTER:   LESSON NEXTLESSON
TO START ANOTHER LESSON, ENTER:   LESSON NNN
      )



       LESSON NEXTLESSON
..............................  ⍴  ..................................
THE SYMBOL ⍴ IS USED FOR THE DYADIC RESHAPE FUNCTION,
   THIS TAKES THE ELEMENTS OF THE RIGHT ARGUMENT AND REARRANGES THEM
   ACCORDING TO THE SHAPE SPECIFIED BY THE LEFT ARGUMENT,
THE LEFT ARGUMENT MUST BE AN INTEGER SCALAR OR A VECTOR OF INTEGER
   ELEMENTS,
THE RIGHT ARGUMENT CAN BE NUMERIC OR CHARACTER, OF ANY RANK,

HOW MANY ARGUMENTS DOES THE RESHAPE FUNCTION TAKE?
.2
RIGHT
```

```
WHAT RANK CAN THE LEFT ARGUMENT OF RESHAPE BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
,V
RIGHT

WHAT RANK CAN THE RIGHT ARGUMENT OF RESHAPE BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
,A
RIGHT

IF THE LEFT ARGUMENT IS A SCALAR, THE RESULT WILL BE A VECTOR WITH
   LENGTH EQUAL TO THE SCALAR, CONSISTING OF THE ELEMENTS OF THE RIGHT
   ARGUMENT, TAKEN IN ORDER TOP LEFT TO BOTTOM RIGHT.
WHEN THERE ARE NOT ENOUGH ELEMENTS IN THE RIGHT ARGUMENT,
   THE FUNCTION WILL START OVER AND TAKE THE ELEMENTS IN ORDER AGAIN,
   AS MANY TIMES AS NECESSARY TO FILL THE VECTOR.
FOR EXAMPLE:   4⍴1 => 1 1 1 1
               4⍴('AB') => ABAB
IF THERE ARE TOO MANY ELEMENTS ON THE RIGHT, THE EXTRAS ARE OMITTED.
FOR EXAMPLE:   2⍴(1 2 3 4) => 1 2

DOES S1⍴S2  EQUAL  S2⍴S1?
,N
RIGHT

WHAT IS THE RESULT OF:
 2 ⍴  0.6
```

```
..6
SORRY, CORRECT ANSWER IS:
0.6 0.6
WHAT IS THE RESULT OF:
 2 ⍴ 0.1
..1 .1 .1
SORRY, CORRECT ANSWER IS:
0.1 0.1
WHAT IS THE RESULT OF:
 3 ⍴ 0.6
..6 .6 .6
CORRECT:

IF THE LEFT ARGUMENT IS A VECTOR, THE ELEMENTS INDICATE, IN ORDER,
   LENGTH OF EACH DIMENSION IN THE RESULT.
FOR EXAMPLE:  2 2⍴1 2 3 4  =>  1 2
                               3 4
THIS IS ONE WAY TO ENTER MATRIXES OR ARRAYS OF HIGHER DIMENSION.

IF ONE ARGUMENT IS A VECTOR, THE OTHER ARGUMENT MAY BE SCALAR OR VECTOR.
       (T/F)
.T
RIGHT

WHAT IS THE RESULT OF:
 3 ⍴ 0.1 4
..1 4 .1
CORRECT:

USING A MATRIX OR HIGHER-LEVEL ARRAY AS THE LEFT ARGUMENT OF RESHAPE
   WILL RESULT IN A 'RANK ERROR'.

IF ONE ARGUMENT IS A MATRIX, THE OTHER ARGUMENT MUST BE A SCALAR
 OR A MATRIX OF THE SAME SHAPE. (T/F)
```

```
,F
RIGHT


CAN RESHAPE TAKE CHARACTER ARGUMENTS?
,T
RIGHT

IF SO, WHICH ARGUMENT(S) CAN BE CHARACTER?
ENTER R FOR RIGHT,
      L FOR LEFT, OR
      B FOR BOTH
,R
RIGHT

THIS IS THE END OF YOUR LESSON.
FOR THE NEXT LESSON IN SEQUENCE, ENTER:   LESSON NEXTLESSON
TO START ANOTHER LESSON, ENTER:   LESSON NNN
      .
```

```
      TEACH
ENTER THE SYMBOL(S) YOU WOULD LIKE INFORMATION ABOUT,
     OR FOR MENU SELECTION, ENTER:  MENU
.+


.................................+.....................................
THE SYMBOL + IS USED FOR THE MONADIC CONJUGATE FUNCTION,
THIS IS AN IDENTITY FUNCTION WHICH RETURNS ITS ARGUMENT, AFTER
   EVALUATION IF NECESSARY,  IT TAKES NUMERIC ARGUMENTS OF ANY RANK,

HOW MANY ARGUMENTS DOES THE CONJUGATE FUNCTION TAKE?
.1
RIGHT

WHAT RANK CAN THE RIGHT ARGUMENT OF CONJUGATE BE?
ENTER S FOR SCALAR,
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
.A
RIGHT
EXAMPLE:  +2  =>  2          +¯3+2  =>  ¯1,



CAN CONJUGATE TAKE CHARACTER ARGUMENTS?
.N
RIGHT


................................ + ....................................
THE SYMBOL + IS USED FOR THE DYADIC PLUS FUNCTION,
```

```
THIS PERFORMS SIMPLE ADDITION ON NUMERIC ARGUMENTS OF ANY RANK.

HOW MANY ARGUMENTS DOES THE PLUS FUNCTION TAKE?
.2
RIGHT

WHAT RANK CAN THE LEFT ARGUMENT OF PLUS BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
.A
RIGHT

WHAT RANK CAN THE RIGHT ARGUMENT OF PLUS BE?
ENTER S FOR SCALAR
      V FOR VECTOR
      X FOR MATRIX
      A FOR ANY RANK
.A
RIGHT

FOR EXAMPLE: 2+2  => 4     SPACING IS NOT IMPORTANT.

DOES S1+S2  EQUAL  S2+S1?
.Y
RIGHT

WHAT IS THE RESULT OF:
 10 +  10
.+
SORRY, THIS ANSWER NOT ACCEPTABLE. PLEASE ENTER A NUMERIC ANSWER.

WHAT IS THE RESULT OF:
```

```
 10 +  10
.20
CORRECT!

TO ADD A SCALAR TO EACH ELEMENT OF A VECTOR, ENTER EITHER S+V OR V+S.
V1+V2 WILL ADD TWO VECTORS ELEMENT BY ELEMENT.  FOR EXAMPLE:
   2+(3 4 5)  => 5 6 7        (3 4 5)+2  => 5 6 7
   (3 4 5)+(5 6 7)  => 8 10 12
IF THE TWO VECTORS BEING ADDED ARE NOT THE SAME LENGTH,
   YOU WILL GET A 'LENGTH ERROR'.

IF ONE ARGUMENT IS A VECTOR, THE OTHER ARGUMENT MAY BE SCALAR OR VECTOR.
       (T/F)
.T
RIGHT

WHAT IS THE RESULT OF:
 0.2 +  0.9 0.3
..2 + .9 .3
SORRY, THIS ANSWER NOT ACCEPTABLE, PLEASE ENTER A NUMERIC ANSWER.

WHAT IS THE RESULT OF:
 0.2 +  0.9 0.3
.1.1 .5
CORRECT!

WHAT IS THE RESULT OF:
 ¯5  4  3 +  ¯5
.STOP

THIS LESSON HAS BEEN HALTED.

THIS IS THE END OF SCHEDULED LESSONS.
TO SELECT MORE LESSONS, ENTER:   TEACH
```

```
      ,TEACH
ENTER THE SYMBOL(S) YOU WOULD LIKE INFORMATION ABOUT,
     OR FOR MENU SELECTION, ENTER:  MENU
,MENU
INFORMATION IS AVAILABLE ON THE FOLLOWING SYMBOLS/TOPICS:
101 ¯ NEGATIVE_NUMBER   102 + CONJUGATE         103 + PLUS
104 - CHANGE_SIGN       105 - MINUS             106 × SIGNUM
107 × TIMES             108 ÷ RECIPROCAL        109 ÷ DIVIDE
121 ( PARENS            122 ' QUOTE             123 ← SPECIFICATION
124 ⍴ SHAPE             125 ⍴ RESHAPE           126 [ BRACKETS
127 , RAVEL             128 , CATENATE          129 ⍳ INDEX_GENERATOR
131 / REDUCTION         132 ⌿ REDUCTION         141 * EXPONENTIAL
142 * POWER             143 ⍟ NATURAL_LOG       144 ⍟ LOGARITHM
145 ○ PI_TIMES          146 ○ GEOMETRIC         147 ! FACTORIAL
148 ! BINOMIAL          149 | MAGNITUDE         151 \ SCAN
152 ⍀ SCAN              161 ⌈ CEILING           162 ⌊ FLOOR
163 ⌈ MAXIMUM           164 ⌊ MINIMUM           165 | RESIDUE
166 ⍋ GRADE_UP          167 ⍒ GRADE_DOWN        168 ? ROLL
169 ? DEAL              180 = EQUAL             181 ≠ NOT_EQUAL
182 < LESS_THAN         183 ≤ LESS_OR_EQUAL     184 ≥ GREATER_OR_EQUAL
185 > GREATER_THAN      191 ~ NOT               192 ∧ AND
193 ⍲ NAND              194 ∨ OR                195 ⍱ NOR
196 ∈ MEMBER OF         201 ⌽ REVERSE           202 ⊖ REVERSE
203 ⌽ ROTATE            204 ⊖ ROTATE            205 ⍉ TRANSPOSE-MONADIC
206 ⍉ TRANSPOSE-DYADIC  211 ↑ TAKE              212 ↓ DROP
213 / COMPRESS          214 ⌿ COMPRESS          215 \ EXPAND
216 ⍀ EXPAND            219 , LAMINATE          231 ⌹ MATRIX_INVERSE
232 ⌹ MATRIX_DIVIDE     241 ∘ OUTER_PRODUCT     242 . INNER_PRODUCT
301 ⍳ INDEX_OF          304 ⍤ ENCODE            305 ⊥ DECODE
306 ⍎ EXECUTE           307 ⍕ FORMAT-DYADIC     308 ⍕ FORMAT-MONADIC
TO SEE MENU AGAIN, ENTER:                MENU
FOR INFORMATION ABOUT FUNCTION, ENTER:   INFO NNN   (WHERE NNN IS MENU N
      UMBER)
TO GO THROUGH TUTORIAL LESSON, ENTER:    LESSON NNN
```

LESSON 105

..... - .....

THE SYMBOL - (UPSHIFT +) IS USED FOR THE DYADIC MINUS FUNCTION.
IT PERFORMS SIMPLE SUBTRACTION ON NUMERIC ARGUMENTS OF ANY RANK.

HOW MANY ARGUMENTS DOES THE MINUS FUNCTION TAKE?

.2

RIGHT

WHAT RANK CAN THE LEFT ARGUMENT OF MINUS BE?

ENTER S FOR SCALAR

V FOR VECTOR

X FOR MATRIX

A FOR ANY RANK

.A

RIGHT

WHAT RANK CAN THE RIGHT ARGUMENT OF MINUS BE?

ENTER S FOR SCALAR

V FOR VECTOR

X FOR MATRIX

A FOR ANY RANK

.A

RIGHT

FOR EXAMPLE: 5-2 => 3 ¯10-¯2 => ¯8

DOES S1-S2 EQUAL S2-S1?

.N

RIGHT

WHAT IS THE RESULT OF:

0.1 - ¯0.3

..4

```
CORRECT!

TO SUBTRACT A SCALAR FROM EACH ELEMENT OF A VECTOR, ENTER V-S,
TO SUBTRACT EACH ELEMENT OF A VECTOR FROM THE SAME SCALAR, ENTER S-V,
V1-V2 WILL SUBTRACT EACH ELEMENT OF V2 FROM THE CORRESPONDING
   ELEMENT OF V1,
V1 AND V2 MUST BE THE SAME LENGTH OR YOU WILL GET A 'LENGTH ERROR',
FOR EXAMPLE:  (3 4 5)-2  => 1 2 3        2-(3 4 5)  => ¯1 ¯2 ¯3
              (6 7 8)-(1 2 3)  => 5 5 5

IF ONE ARGUMENT IS A VECTOR, THE OTHER ARGUMENT MAY BE SCALAR OR VECTOR,
       (T/F)
.T
RIGHT

WHAT IS THE RESULT OF:
 0.9 -  0.6 0.5
..3 .4
CORRECT!

WHAT IS THE RESULT OF:
 1 2 3 -  ¯5
.¯4 ¯3 ¯2
SORRY, CORRECT ANSWER IS:
6 7 8
WHAT IS THE RESULT OF:
 ¯0.9  0.4 -  ¯0.4
.¯.5 .8
CORRECT!

WHAT IS THE RESULT OF:
 ¯0.7  0.8 -  ¯0.8  0.7
..1 1.5
SORRY, CORRECT ANSWER IS:
```

```
0.1 0.1
WHAT IS THE RESULT OF:
  80 ¯60  50 ¯80 -   90  30 ¯10 ¯50
.¯10 ¯90 60 ¯30
CORRECT:

ENTER M-S TO SUBTRACT THE SCALAR S FROM EACH ELEMENT OF THE MATRIX M.
ENTER S-M TO DO THE OPPOSITE.
FOR EXAMPLE: IF M => 1 2  THEN 4-M  =>  3 2  AND  M-2  => ¯1 0
                    3 4                 1 0                1 2
M1-M2 SUBTRACTS MATRIX M2 FROM MATRIX M1, ELEMENT BY ELEMENT.
FOR EXAMPLE: IF  M1 => 1 2  AND  M2 => 3 4  THEN M2-M1  =>  2 2
                       3 4             5 6                 2 2
IF M1 AND M2 ARE NOT THE SAME SHAPE, A 'LENGTH ERROR' WILL RESULT.
TRYING TO SUBTRACT A VECTOR FROM A MATRIX OR VICE-VERSA WILL
   PRODUCE A 'RANK ERROR'.

IF ONE ARGUMENT IS A MATRIX, THE OTHER ARGUMENT MUST BE A SCALAR
 OR A MATRIX OF THE SAME SHAPE. (T/F)
.T
RIGHT

THESE PRINCIPLES CAN BE EXTENDED TO HIGHER LEVEL ARRAYS.

CAN MINUS TAKE CHARACTER ARGUMENTS?
.N
RIGHT

WRITE A TRUE STATEMENT USING -.
FOR EXAMPLE: 4=2+2
.2-2=0
SORRY, THIS STATEMENT IS NOT TRUE.
TRY AGAIN.
```

```
WRITE A TRUE STATEMENT USING -.
FOR EXAMPLE: 4=2+2
.4=2+2
YOUR ANSWER DOES NOT USE -. TRY AGAIN.

WRITE A TRUE STATEMENT USING -.
FOR EXAMPLE: 4=2+2
.(2-2)=4

DO YOU WISH TO REVIEW THE DESCRIPTION OF MINUS?
ENTER Y OR N.
.N

WRITE A TRUE STATEMENT USING -.
FOR EXAMPLE: 4=2+2
.0=2-2
CORRECT!

THIS IS THE END OF YOUR LESSON.
FOR THE NEXT LESSON IN SEQUENCE, ENTER:   LESSON NEXTLESSON
TO START ANOTHER LESSON, ENTER:   LESSON NNN
      .
```

```
      HELP
ENTER THE SYMBOL(S) YOU WOULD LIKE INFORMATION ABOUT,
   OR...FOR MENU SELECTION, ENTER:  MENU
.⍳

..................................⍳.....................................
THE SYMBOL ⍳ (UPSHIFT I) IS USED FOR THE MONADIC INDEX GENERATING
   FUNCTION,
IT IS USED WITH A SINGLE NON-NEGATIVE INTEGER ARGUMENT (SCALAR, OR
   VECTOR OF LENGTH 1),
IT RETURNS A VECTOR OF INTEGERS, IN ORDER, BEGINNING WITH THE INDEX
   ORIGIN, AND ENDING WITH THE ARGUMENT,
FOR EXAMPLE:    ⍳3  ⇒  1 2 3       ⍳0  ⇒      (AN EMPTY VECTOR)
THE INDEX ORIGIN IS NORMALLY (BY DEFAULT) 1,
USING ⍳ WITH A NON-INTEGER, NEGATIVE OR ARRAY ARGUMENT WILL PRODUCE A
   'DOMAIN ERROR',



..................................⍳.....................................
THE SYMBOL ⍳ (UPSHIFT I) IS USED FOR THE DYADIC INDEX OF FUNCTION,
THE LEFT ARGUMENT MUST BE A VECTOR AND THE RIGHT ARGUMENT CAN BE
   ANY RANK,  INDEX OF CAN BE USED WITH CHARACTER OR NUMERIC DATA,
IT RETURNS THE POSITION IN THE LEFT ARGUMENT OF THE ELEMENTS OF THE
   RIGHT ARGUMENT,  THE SHAPE OF THE RESULT IS THE SAME AS THE
   SHAPE OF THE RIGHT ARGUMENT,
IF THE RIGHT ARGUMENT DOES NOT OCCUR IN THE LEFT ARGUMENT, THE RESULT
   WILL BE ONE PLUS THE LENGTH OF THE LEFT ARGUMENT,

FOR EXAMPLE:  4 3 2 1⍳3   ⇒   2   BECAUSE 3 IS IN THE SECOND POSITION
              'HELLO'⍳'P' ⇒   6   BECAUSE P IS NOT FOUND

MORE?
```

```
.T

V\A CHECKS EACH ELEMENT OF THE RIGHT ARGUMENT INDIVIDUALLY.
FOR EXAMPLE:  3 5 7 9\1 2 3 4 5  =>  5 5 1 5 2
   IF YOU ASSIGN   ALPHABET←'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
   THEN ALPHABET\'APL'  =>  1 16 12


FOR MORE HELP, ENTER:   HELP

      .
```

```
      MENU
INFORMATION IS AVAILABLE ON THE FOLLOWING SYMBOLS/TOPICS;
101 ¯ NEGATIVE_NUMBER   102 + CONJUGATE         103 + PLUS
104 - CHANGE_SIGN       105 - MINUS             106 × SIGNUM
107 × TIMES             108 ÷ RECIPROCAL        109 ÷ DIVIDE
121 ( PARENS            122 ' QUOTE             123 ← SPECIFICATION
124 ⍴ SHAPE             125 ⍴ RESHAPE           126 [ BRACKETS
127 , RAVEL             128 , CATENATE          129 ⍳ INDEX_GENERATOR
131 / REDUCTION         132 ⌿ REDUCTION         141 * EXPONENTIAL
142 * POWER             143 ⍟ NATURAL_LOG       144 ⍟ LOGARITHM
145 ○ PI_TIMES          146 ○ GEOMETRIC         147 ! FACTORIAL
148 ! BINOMIAL          149 | MAGNITUDE         151 \ SCAN
152 ⍀ SCAN              161 ⌈ CEILING           162 ⌊ FLOOR
163 ⌈ MAXIMUM           164 ⌊ MINIMUM           165 | RESIDUE
166 ⍋ GRADE_UP          167 ⍒ GRADE_DOWN        168 ? ROLL
169 ? DEAL              180 = EQUAL             181 ≠ NOT_EQUAL
182 < LESS_THAN         183 ≤ LESS_OR_EQUAL     184 ≥ GREATER_OR_EQUAL
185 > GREATER_THAN      191 ~ NOT               192 ∧ AND
193 ⍲ NAND              194 ∨ OR                195 ⍱ NOR
196 ∈ MEMBER OF         201 ⌽ REVERSE           202 ⊖ REVERSE
203 ⌽ ROTATE            204 ⊖ ROTATE            205 ⍉ TRANSPOSE-MONADIC
206 ⍉ TRANSPOSE-DYADIC  211 ↑ TAKE              212 ↓ DROP
213 / COMPRESS          214 ⌿ COMPRESS          215 \ EXPAND
216 ⍀ EXPAND            219 , LAMINATE          231 ⌹ MATRIX_INVERSE
232 ⌹ MATRIX_DIVIDE     241 ∘ OUTER_PRODUCT     242 . INNER_PRODUCT
301 ⍳ INDEX_OF          304 ⊤ ENCODE            305 ⊥ DECODE
306 ⍎ EXECUTE           307 ⍕ FORMAT-DYADIC     308 ⍕ FORMAT-MONADIC
TO SEE MENU AGAIN, ENTER;                 MENU
FOR INFORMATION ABOUT FUNCTION, ENTER;   INFO NNN   (WHERE NNN IS MENU N
      UMBER)
TO GO THROUGH TUTORIAL LESSON, ENTER;    LESSON NNN
      •
```

```
      INFO 180

..................................  =  ..................................
THE SYMBOL = (UPSHIFT 5) IS USED FOR THE DYADIC EQUALS FUNCTION,
THIS COMPARES NUMERIC OR CHARACTER ARGUMENTS OF ANY RANK,
IT RETURNS 1 FOR EACH ELEMENT OF THE LEFT ARGUMENT THAT IS IDENTICAL
   TO THE CORRESPONDING ELEMENT OF THE RIGHT ARGUMENT, AND 0 FOR
   EACH ELEMENT THAT IS NOT,

FOR EXAMPLE: 2=2  => 1     2=4   => 0       'A'='B'   => 0

MORE?
,N

FOR MORE INFORMATION, ENTER:  INFO NNN
      .
```

```
       TEST 211

WHAT IS THE RESULT OF:
 ¯3 ↑  ¯30  40  60 ¯90
.40 60 ¯90
CORRECT!
TEST OK
      .

       TEST 203

WHAT IS THE RESULT OF:
 ¯1 ⌽   60  70 ¯20 ¯40
.
SORRY, CORRECT ANSWER IS:
¯40 60 70 ¯20
WHAT IS THE RESULT OF:
 2 ⌽    0 ¯60 ¯40 ¯10
.¯40 ¯10 0 ¯60
CORRECT!
TEST OK
      .

       TEST 152

WHAT IS THE RESULT OF:
 ⌈↖  7 ¯6 ¯1
.7 7 7
CORRECT!
TEST OK
      .
```

```
      TEST 166

WHAT IS THE RESULT OF:
 ⍋  ¯0.3 ¯0.4
.2 1
CORRECT:

WHAT IS THE RESULT OF:
 ⍋   3  5 ¯6
.STOP

THIS TEST HAS BEEN HALTED.
      .



       STOP
IF YOU WANT TO END THIS SESSION, ENTER:     )SAVE
WHEN YOU SEE THE TIME, DATE, AND 'TUTOR' MESSAGE, ENTER:    )OFF
      .)SAVE
11:42:19 12/14/83 TUTOR
      .)OFF
CONNECT= 01:10:27 VIRTCPU= 000:02.88 TOTCPU= 000:11.11
CONNECT=  $5.87 TOTCPU=  $1.85 SIO=  $0.41  TOTAL=   $8.13
MULT BY SHIFT FACTOR: =1(DAY), =0.6(EVE), =0.3(NIGHTS)
LOGOFF AT 11:42:42 PST WEDNESDAY 12/14/83

VM/370 ONLINE
```

BIBLIOGRAPHY

Gilman, Leonard, and Rose, Allen J. APL--An Interactive Approach. 2nd ed. New York: John Wiley & Sons, Inc., 1974.

Ramsey, James B., and Musgrave, Gerald L. APL-STAT: A Do-it-yourself Guide to Computational Statistics Using APL. Belmont, California: Lifetime Learning Publications, 1981.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Dennis R. Mar, Code 0141 Naval Postgraduate School Monterey, California 93943	1
4.	LT Katherine S. Lanes c/o Deputy Commander Operational Test and Evaluation Force, Pacific Naval Air Station North Island San Diego, California 92135	1
5.	Professor R. R. Read, Code 55Re Naval Postgraduate School Monterey, California 93943	1

207129

Thesis
L2592 Lanes
c.1 APL TUTOR: an on-
line instructional fa-
cility.